

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การพัฒนาแอปพลิเคชันด้วยเว็บเซอร์วิส

APPLICATION DEVELOPMENT WITH WEB SERVICE

นายกฤษณ์ วรสุวรรณรักษ์
นายกฤษดา วัฒนาอรพงศ์
นายชลธรรม ประภากรวรกุล

รฟ.
๗๒๘๑๖
๒๕๔๘

เลขหมู่.....
เลขทะเบียน..... 62340
วัน,เดือน,ปี..... 16 ส.ค. 2549

b..... 1162031x
i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

การพัฒนาแอปพลิเคชันด้วยเว็บเซอร์วิส
APPLICATION DEVELOPMENT WITH WEB SERVICE



โดย
นายกฤษฎ์ วรสุวรรณรักษ์
นายกฤษดา วัฒนาจารุพงศ์
นายชลธรรม ประภากรวรกุล

อาจารย์ที่ปรึกษา
ดร. วรวัฒน์ ลิ้มโกศา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาแอปพลิเคชันด้วยเว็บเซอร์วิส

APPLICATION DEVELOPMENT WITH WEBSERVICE

ผู้จัดทำ

1. นายกฤษณ์ วรสุวรรณรักษ์ รหัสนักศึกษา 45010020
2. นายกฤษดา วัฒนาจารุพงศ์ รหัสนักศึกษา 45010028
3. นายชลธรรม ประภากรวรกุล รหัสนักศึกษา 45010160



อาจารย์ที่ปรึกษา

(วรวัฒน์ ลิมโกตา)



การพัฒนาแอปพลิเคชันด้วยเว็บเซอร์วิส

นายกฤษณ์	วรสุวรรณรักษ์	45010020
นายกฤษดา	วัฒนาจารุพงศ์	45010028
นายชลธรรม	ประภากรวรกุล	45010160
ดร.วรวัฒน์	ลิ้มโกศา	อาจารย์ที่ปรึกษา
ปีการศึกษา 2548		

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เสนอวิธีการพัฒนาเว็บเซอร์วิสบนจาวาแพลตฟอร์มและนำระบบความปลอดภัยไปใช้ในเว็บเซอร์วิสซึ่งทำให้เว็บเซอร์วิสมีความปลอดภัยและน่าเชื่อถือตามมาตรฐานของ WS-Security และ WS-Interoperability ซึ่งมีความจำเป็นในปัจจุบันซึ่งมีเว็บเซอร์วิสเกิดขึ้นมากมายเพื่อให้บริการแก่ลูกค้าในรูปแบบต่างๆ โดยนำเสนอในส่วนของการส่งข้อมูลที่มีความสำคัญระหว่างแพลตฟอร์มเดียวกัน คือ Java กับ Java และ โดยใช้ IBM Websphere Application Developer 5.1.1 และ ตัวแพลตฟอร์มคือ ระหว่าง Java และ Microsoft Visual Studio .Net 2003 with WSE 1.0 พร้อมทั้งวิธีการติดตั้ง รวมถึงอธิบายองค์ประกอบอื่นๆของเว็บเซอร์วิส และ ทดสอบโดยการนำไปใช้ในระบบตัวอย่างของ OLALA EDUCATION ซึ่งพัฒนาให้ระบบใช้เว็บเซอร์วิสและนำหลักการของ WS-Security และ WS-Interoperability ไปใช้จริง

Application Development using Java Web Service

Krit Worasuwannarak 45010020

Krisda Wattanajarupong 45010028

Sholltham Praparkornvorakul 45010160

Mr. Vorawat Limpoka Advisor

Academic Year 2005

Abstract

This thesis methods of building Java Web Services which implement Web Service Security and Web Service Interoperability .This thesis also mention about how to implement Security in Java Web Service and how can Java Web Service implement WS-Security using IBM Websphere Application Developer 5.1.1 interoperate with Microsoft .Net with WSE1.0 . And also test in sample Application “OLALA EDUCATION” ,which implement WS-Security in Java and WSE1.0 in Microsoft .Net

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้คงไม่อาจสำเร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและความร่วมมือจากบุคคลดังนี้ ลำดับแรกขอขอบคุณ ดร. วรวัฒน์ ลิ้มโกศา ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ที่คอยให้คำปรึกษาและแนะนำแนวทาง ข้าพเจ้ารู้สึกทราบบ้างในความอนุเคราะห์ ขอขอบพระคุณเป็นอย่างสูง

ขอกราบพระคุณณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ลำดับต่อมาคือ บริษัท IBM ซึ่งได้คอยช่วยให้คำแนะนำเกี่ยวกับการพัฒนาแอปพลิเคชันและ เว็บเซอร์วิส

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจ และให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอบแต่ผู้มีพระคุณทุกท่าน

นายกฤษณ์	วรสุวรรณ์รักษ์
นายกฤษดา	วัฒนาจารุพงศ์
นายชลธรรม	ประภากรวรกุล

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ.....	IV
สารบัญตาราง.....	VIII
สารบัญรูป.....	IX
บทที่ 1 บทนำ.....	1
1.1. ความสำคัญและที่มาของโครงการ	1
1.2. วัตถุประสงค์ของโครงการ	2
1.3. ขอบเขตของโครงการ	2
1.4. วิธีการดำเนินงาน	2
บทที่ 2 ทฤษฎีพื้นฐานเกี่ยวกับเว็บเซอร์วิส	3
2.1. เว็บเซอร์วิส.....	3
2.1.1. ประโยชน์ของการทำงานร่วมกันระหว่าง แอปพลิเคชัน กับ เว็บเซอร์วิส	3
2.1.2. การทำงานของเว็บเซอร์วิส	4
2.1.3. เปรียบเทียบระหว่าง Web Application และ Web Services	5
2.1.4. การประยุกต์ใช้งานเว็บเซอร์วิส	6
2.2. Service-Oriented Architecture (SOA)	6
2.3. XML.....	8
2.3.1. XML & HTML.....	9
2.4. SOAP (Simple Object Access Protocol).....	10
2.4.1. ทำไมต้องเป็น SOAP	11
2.4.2. โครงสร้างของ SOAP	12
2.5. WSDL (Web Services Description Language).....	17
2.5.1. โครงสร้างเอกสาร WSDL	17
2.5.2. WSDL Example	19
2.6. UDDI (Universal Description, Discovery and Integration protocol)	25

สารบัญ (ต่อ)

	หน้า
บทที่ 3 Web Service Security (WS-Security).....	26
3.1. Web Service Security (WS-Security)	26
3.2. การเพิ่มความปลอดภัยในการพิสูจน์ตัวตน (Authentication)	27
3.3. การรักษาความสมบูรณ์ของข้อมูล (Integrity)	29
3.4. การเพิ่มความปลอดภัยของข้อมูลที่เป็นความลับ (Confidentiality)	32
3.4.1. อัลกอริทึมในการเข้ารหัสข้อมูลมี 2 ประเภทหลัก คือ	32
3.4.1.1. อัลกอริทึมแบบสมมาตร (Symmetric key algorithms).....	32
3.4.1.2. อัลกอริทึมแบบอสมมาตร (Asymmetric key algorithms).....	32
3.4.2. ปัญหาของอัลกอริทึมแบบสมมาตร(Symmetric key Problem).....	32
3.4.3. ความแข็งแกร่งของอัลกอริทึมสำหรับการเข้ารหัส	34
3.4.4. ความยาวของกุญแจที่ใช้ในการเข้ารหัส	35
3.4.5. อัลกอริทึมสำหรับการเข้ารหัสแบบสมมาตร	35
3.4.6. อัลกอริทึมสำหรับการเข้ารหัสแบบกุญแจสาธารณะ	36
3.4.7. อัลกอริทึมสำหรับสร้างเมสเสจไดเจสต์	36
3.4.8. คุณสมบัติที่สำคัญของอัลกอริทึมสำหรับสร้าง ไดเจสต์มีดังนี้.....	37
3.5. การเพิ่มความปลอดภัยในการขนส่งข้อมูล (Transport – level Security).....	39
3.5.1. หลักการทำงานของ SSL.....	40
3.6. รูปแบบของคีย์และการนำไปใช้	43
3.7. Certificates.....	46
3.7.1. ส่วนประกอบของ Certificate	46
3.7.2. การใช้ CA ภายในหรือภายนอก.....	47
3.7.3. Certification Authority (CA).....	47
3.7.4. การสร้าง Self-signed Certificate	48
3.7.5. Registration Authority (RA).....	49
3.7.6. Certificate Management Protocol (CMP)	49
3.7.7. Certificate Revocation	50
3.7.8. Certificate Repository	51
3.7.9. Time Stamp Authority (TSAs).....	51

สารบัญ (ต่อ)

	หน้า
3.7.10. แผนโครงสร้างของ OpenCA.....	51
3.7.11. แผนโมเดลการให้ความเชื่อถือ (Trust Model)	52
3.7.12. เข้าใจหน้าที่ของ PKI	53
3.8. การออกแบบ Public Key Infrastructure.....	54
3.8.1. กำหนดความต้องการ Certificate	54
3.8.2. การสร้างโครงสร้างพื้นฐาน Certification.....	55
3.8.2.1. การสร้าง CA เป็นลำดับชั้น	56
3.8.2.2. ชนิดของ Windows Server 2003 CA.....	57
3.8.3. การกำหนดค่า Certification	57
3.9. การจัดการ Certificates.....	57
3.10. KeyStore ในภาษาจาวา และ ดอทเน็ต.....	60
บทที่ 4 การทดลองและผลการทดลอง	63
4.1. โครงร่างของระบบ	63
4.1.1. ฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในการทดลอง	63
4.1.2. แบบจำลองระบบที่มีการนำ WS-Security ไปใช้.....	65
4.2. ผลการทดลอง.....	66
4.2.1. การสร้างสียสำหรับใช้ในเว็บเซอร์วิสด้วย Java Toolkit 1.03	66
4.2.2. การเซตค่าเพื่อให้สอดคล้องกับหลักการ WS-Security ระหว่าง Java Web service และ Java Webservice โดยใช้ Websphere Application Developer 5.1.1.....	68
4.2.2.1. การเพิ่มความปลอดภัยในการพิสูจน์ตัวตน(Authentication)	68
4.2.2.2. การรักษาความสมบูรณ์ของข้อมูล (Integrity)	70
4.2.2.3. การเพิ่มความปลอดภัยของข้อมูลที่เป็นความลับ (Confidentiality)	72
4.2.2.3.1. การเข้ารหัสทั้งเอกสาร	74
4.2.2.3.2. การเข้ารหัสเฉพาะ Element	74
4.2.2.3.3. การเข้ารหัสเฉพาะข้อมูลในเอกสาร	75
4.2.2.4. การเพิ่มความปลอดภัยในการขนส่งข้อมูล (Transport – level Security).....	76
4.2.3. การเซตค่าเพื่อใช้ในการทำ WS-Security แบบ Encryption	77
4.2.3.1 การเซตค่าของ Websphere Application Developer 5.1.....	77

สารบัญ (ต่อ)

	หน้า
4.2.3.1.1 การสร้างเว็บเซอร์วิส	77
4.2.3.2 การเซตค่าของ Microsoft Visual Studio .Net 2003 with WSE 1.0	83
4.2.3.2.1 การ install key	83
4.2.3.2.2. การสร้างเว็บแอปพลิเคชัน	85
4.2.4. การเซตค่าเพื่อใช้ในการทำ WS-Security	88
4.2.4.1. การเซตค่าของ Websphere Application Developer 5.1	88
4.2.4.2. การเซตค่าของ Visual Studio .Net 2003	89
4.2.5. SSL	95
4.2.5.1 การติดตั้ง SSL บน IIS สำหรับใช้ใน ASP.Net Web Application	95
4.3. การทดลองWS-Security กับ เครื่องมืออื่นๆ ดังนี้	100
4.5. เปรียบเทียบการทำงานของเว็บเซอร์วิสจากและ ไมโครซอฟต์คอตเน็ต	102
บทที่ 5 บทวิจารณ์และสรุป	104
5.1 บทวิจารณ์และสรุป	104
5.2. แนวทางในการพัฒนาต่อ	105
บรรณานุกรม	106

สารบัญตาราง

หน้า

ตารางที่ 2.1 แสดงการเปรียบเทียบระหว่าง เว็บแอปพลิเคชันกับเว็บเซอร์วิส.....	5
ตารางที่ 2.2 แสดงส่วนประกอบของ SOAP	14
ตารางที่ 2.3 อธิบาย Fault Element.....	15
ตารางที่ 2.4 แสดง Fault code Error	16
ตารางที่ 2.5 แสดง WSDL ELEMENT	19
ตารางที่ 3.1 แสดง การเปรียบเทียบความสามารถของกีย์แต่ละตัว.....	43
ตารางที่ 3.2 แสดง การเปรียบเทียบระดับความปลอดภัย	55
ตารางที่ 3.3 แสดง การเปรียบเทียบระหว่าง CA ภายในกับ CA ภายนอก.....	55
ตารางที่ 4.1 แสดงรูปแบบการสร้าง กีย์แบบต่างๆ	67



สารบัญรูป

หน้า

รูปที่ 2.1 อธิบายการทำงานของเว็บเซอร์วิส.....	4
รูปที่ 2.2 แสดงส่วนประกอบและการทำงานของ SOA	7
รูปที่ 2.3 แสดงการติดต่อกันของส่วนประกอบหลัก.....	8
รูปที่ 2.4 แสดง Tag ของภาษา XML.....	9
รูปที่ 2.5 แสดง XML messaging using SOAP	11
รูปที่ 2.6 โครงสร้างของเอกสาร SOAP	12
รูปที่ 2.7 ตัวอย่างเอกสาร SOAP อย่างง่าย	12
รูปที่ 2.8 ตัวอย่างเอกสาร SOAP	13
รูปที่ 2.9 ตัวอย่างข้อมูลใน <Fault> Element ที่มีการเกิดข้อผิดพลาด	15
รูปที่ 2.10 แสดง ส่วนประกอบของ WSDL ในรูปแบบ XML.....	17
รูปที่ 2.11 แสดง ส่วนประกอบ WSDL ในรูปแบบโครงสร้าง.....	18
รูปที่ 2.12 แสดงการทำงานของ UDDI	25
รูปที่ 3.1 แสดงตัวอย่างการส่งข้อมูลที่ไม่มีการจัดการเรื่อง Security	26
รูปที่ 3.2 แสดงโมเดล Securing Web Services	27
รูปที่ 3.3 แสดงประเภทของ Token	28
รูปที่ 3.4 แสดงตัวอย่าง SOAP ที่มีการทำ Username Token แล้ว	29
รูปที่ 3.5 การส่งข้อมูลเข้าไปใน Hash function	30
รูปที่ 3.6 การเข้ารหัสแบบไฮบริดด้วยกุญแจส่วนตัวเพื่อเป็นการลงลายเซ็น	30
รูปที่ 3.7 ขั้นตอนการเปรียบเทียบความถูกต้อง	30
รูปที่ 3.8 แสดงตัวอย่าง SOAP Message ที่มีการใช้ Digital signature.....	31
รูปที่ 3.9 แสดงตัวอย่าง SOAP Message มีการเข้ารหัสทั้งเอกสารแบบ triple-DES.....	39
รูปที่ 3.10 กระบวนการเริ่มต้นการติดต่อสื่อสารของโพรโตคอล SSL	41
รูปที่ 3.11 แสดงกระบวนการสร้าง Self-signed Certificate	48
รูปที่ 3.12 แสดงโครงสร้างของ OpenCA	52
รูปที่ 3.13 แสดง Trust Model แบบ Hierarchy.....	52
รูปที่ 3.14 แสดง Trust Model แบบ Cross-Certification.....	53
รูปที่ 4.1 แสดงแบบจำลองระบบที่มีการนำ WS-Security ไปใช้	65
รูปที่ 4.2 แสดง โปรแกรม Java Toolkit	66

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่ 4.3 แสดงการสร้างคีย์ด้วยโปรแกรม Java Toolkit	66
รูปที่ 4.4 แสดงผลการสร้างคีย์ด้วยโปรแกรม Java Toolkit เป็น command	67
รูปที่ 4.5 แสดง ตัวอย่างโปรแกรม Echo Service ที่มีการทำ Username Token.....	68
รูปที่ 4.6 แสดง SOAP ที่ยังไม่ได้มีการทำ Username Token	69
รูปที่ 4.7 แสดง SOAP ที่มีการทำ Username Token แล้ว.....	69
รูปที่ 4.8 แสดงตัวอย่างโปรแกรม Echo ที่มีการทำ Digital Signature แล้ว	70
รูปที่ 4.9 แสดง SOAP Message ที่ไม่มีการ ทำ Digital signature.....	70
รูปที่ 4.10 แสดงSOAP Message ที่มีการ ใช้ Digital signature	71
รูปที่ 4.11 แสดง การ ส่ง Echo Message ที่มีการ Encryption SOAP เข้าไปด้วย.....	72
รูปที่ 4.12 แสดง การ ส่ง Echo Message ที่มีการ Encryption SOAP เข้าไปด้วย.....	73
รูปที่ 4.13 แสดง ตัวอย่าง XML ที่มีหลาย element.....	73
รูปที่ 4.14 แสดง SOAP Message มีการเข้ารหัสทั้งเอกสารแบบ triple-DES	74
รูปที่ 4.15 แสดงการเข้ารหัส เฉพาะ Element < Payment > แบบ tripledes.....	74
รูปที่ 4.16 แสดงการเข้ารหัสเฉพาะข้อมูลใน < CardNo > แบบ tripledes.....	75
รูปที่ 4.17 แสดง Echo Service ที่มีการ Configure SSL ที่ พอร์ต 9443.....	76
รูปที่ 4.18 แสดง Certificate และ แสดงว่า มีการติดตั้ง SSL บน Server เรียบร้อยแล้ว	77
รูปที่ 4.19 แสดง การสร้างเว็บเซอร์วิส.....	78
รูปที่ 4.20 แสดงการ import ไฟล์จาวา เพื่อใช้ในการทำ เว็บเซอร์วิส.....	79
รูปที่ 4.21 แสดงการสร้างเว็บเซอร์วิส.....	80
รูปที่ 4.22 แสดงข้อมูลในไฟล์ webservice.xml	81
รูปที่ 4.23 แสดงการConfigure ใน security extension.....	81
รูปที่ 4.24 แสดงการ configure Key ที่ใช้ในการ Encryption	82
รูปที่ 4.25 แสดงการ configure encryption.....	83
รูปที่ 4.26 แสดงการ เพิ่มคีย์.....	83
รูปที่ 4.27 แสดง Store ใน Windows.....	84
รูปที่ 4.28 แสดง Certificate ที่ import เข้ามาแล้ว.....	84
รูปที่ 4.29 แสดงการ Add reference Class ของ WSE 1.0	85
รูปที่ 4.30 แสดง Add Web reference ของ Web Service โดยระบุ WSDL.....	85

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่ 4.31 ออกแบบหน้าตาของเว็บให้สอดคล้องกับเว็บเซอร์วิส	86
รูปที่ 4.32 เพิ่ม code ของโปรแกรม เพื่อใช้ในการ encryption	86
รูปที่ 4.33 แสดง SOAP Message ที่มีการใช้ WS-Security	87
รูปที่ 4.34 แสดง require integrity	88
รูปที่ 4.35 แสดงการระบุคีย์ที่ต้องการใช้ใน signature	88
รูปที่ 4.36 แสดง certificate ที่ใช้ trust	89
รูปที่ 4.37 แสดง การ configure ใน signing information	89
รูปที่ 4.38 แสดงการ import key interops-disg.p12	90
รูปที่ 4.39 แสดงผลการ import key ใน personal store	90
รูปที่ 4.40 แสดงการ import key pfx ที่มี private key ต้องมีการใส่รหัสผ่านด้วย	91
รูปที่ 4.41 ทั้ง 2 คีย์ต้องทำใส่ใน personal Store & Trust Root	91
รูปที่ 4.42 แสดงผลการ import ทั้ง 2 สำเร็จ	92
รูปที่ 4.43 แสดง key และ certificate ที่ import เข้าไปแล้ว	92
รูปที่ 4.44 แสดง code ที่เพิ่มเข้าไปให้สามารถทำได้ ทั้ง Digital signature & Encryption	93
รูปที่ 4.45 แสดง SOAP Message Encryption and Digital Signature	94
รูปที่ 4.46 แสดง การเข้าไปจัดการ ใน IIS Manager	95
รูปที่ 4.47 แสดงการเลือกเว็บไซต์	95
รูปที่ 4.48 แสดง properties ในเว็บต่างๆ	96
รูปที่ 4.49 แสดงการ ติดตั้ง SSL	97
รูปที่ 4.50 แสดง certification ที่จะนำไปใช้ในการ SSL	97
รูปที่ 4.51 แสดงให้เห็นว่าเมื่อมีการ configure SSL แล้วจะไม่สามารถเข้าเว็บผ่าน http protocol ได้	98
รูปที่ 4.52 แสดงว่ามีการใช้ SSL ใน https protocol	99
รูปที่ 4.53 จะเห็นว่า Rational & Websphere มีส่วนคล้ายกันมาก แต่สนับสนุน Java ต่างเวอร์ชันกัน	101
รูปที่ 4.54 ส่วนที่เพิ่มเข้ามาใน Rational 6.0 คือ Token Generator ซึ่งไม่มีในเวอร์ชันอื่นๆ	102

บทที่ 1

บทนำ

1.1. ความสำคัญและที่มาของโครงการ

เทคโนโลยีในการกระจายข่าวสารข้อมูลทาง Internet ในปัจจุบันก็คือ เว็บเพจ แต่จากการที่มันมีความสามารถที่จะทำงานได้ด้วยการรวมภาษาทั้ง Client และ Server Side Script ไว้ในตัวเองเช่นภาษา VBScript, Java Script หรือ ASP, PHP, JSP นั้นทำให้เว็บเพจมีลักษณะคล้าย Application จึงถูกเรียกรวมกันว่า Web Application

Web Application สามารถตอบสนองความคิด Distributed Processing ได้ในระดับหนึ่งซึ่งก็คือการแบ่งการประมวลผลไว้ที่ฝั่ง Client และฝั่ง Server และมักจะมีการใช้ database ควบคู่กับการทำ Web Application ไปด้วยตามความต้องการในการทำ E-Business และ E-Commerce ที่กำลังเป็นที่นิยมในปัจจุบัน และเกิดปัญหาที่ตามมาคือ เรื่องของการจ่ายเงินหรือที่เรียกว่า E-payment หรือ Payment-Gateway ซึ่ง Web Application ที่ทำ E-Commerce ต้องใช้บริการจากธนาคาร Online ในการจัดเก็บเงินกับลูกค้า เพราะด้วยเทคโนโลยีนี้การให้บริการเก็บเงินจากธนาคาร Online จำเป็นที่ผู้ค้าต้องไปทำการตกลงกับธนาคารและเขียนโปรแกรมให้ตรงตามมาตรฐานที่ธนาคาร Online กำหนดไว้

ด้วยปัญหายุ่งยากในการค้นหา ติดต่อและตกลงในการขอใช้บริการเก็บเงินจากธนาคาร Online แนวคิด เว็บเซอร์วิสจึงดูเหมือนเป็นทางออกของปัญหานี้ ความเด่นของเทคโนโลยี เว็บเซอร์วิส นี้ก็คือการทำให้ Web กับ Web สามารถติดต่อสื่อสารกันได้ด้วยเอกสาร XML ที่ทั้งคนและคอมพิวเตอร์เข้าใจ และคอมพิวเตอร์ยังสามารถนำข้อมูลนั้นไปประมวลผลต่อได้ ด้วยเอกสาร XML นี้เองทำให้ Web สามารถส่งข้อมูลที่จำเป็นไปให้อีก Web หนึ่งทำงานบางอย่างให้หรือใช้บริการนั่นเอง ทำให้เป็นการง่ายที่จะเขียนโปรแกรมที่จะติดต่อสื่อสารหรือขอใช้บริการเก็บเงินจากธนาคาร Online แต่สำหรับ Web Application ที่ใช้การส่งข้อมูลเป็น html ทำให้ข้อมูลนั้นไม่สามารถนำไปใช้ต่อได้ การเขียนโปรแกรมจึงยุ่งยากตามที่กล่าวด้านบน

แนวคิดของ เว็บเซอร์วิส ก็คือ Web ที่สามารถทำงานอะไรบางอย่างหรือก็คือให้บริการบางอย่างจากการร้องขอจากต่าง Server ด้วยเหตุนี้ทำให้เทคโนโลยี เว็บเซอร์วิส เชื่อมต่อแนวคิด Distributed Processing มากกว่า Web Application และเมื่อประกอบกับการที่ เว็บเซอร์วิส มี UDDI ทำให้เว็บเซอร์วิส สามารถค้นหาบริการต่างๆ ที่ต้องการได้จากทั่วทุกมุมโลก

1.2. วัตถุประสงค์ของโครงการ

- 1.2.1. เพื่อศึกษาการทำงานของเว็บเซอร์วิส และ ส่วนประกอบต่างๆ
- 1.2.2. เพื่อศึกษาแนวทางในการนำเว็บเซอร์วิสไปประยุกต์ใช้กับสถานการณ์ในปัจจุบัน
- 1.2.3. เพื่อศึกษาและเปรียบเทียบการทำงานของจาวาเว็บเซอร์วิส กับเว็บเซอร์วิสอื่นๆ
- 1.2.4. เพื่อศึกษาเกี่ยวกับรายละเอียดและความปลอดภัยของข้อมูลและการติดต่อระหว่างเว็บ

เซอร์วิส

1.3. ขอบเขตของโครงการ

โครงการนี้เน้นศึกษาเกี่ยวกับเว็บเซอร์วิส ในเรื่องของการสร้างเว็บเซอร์วิสด้วย IBM Websphere Application Developer version 5.1.1 พร้อมทั้งการจัดการเกี่ยวกับความปลอดภัยของ Web service (WS-Security) และการเชื่อมต่อกันระหว่าง ASP .Net Web service และ Java Web service โดยมีส่วนของ ความปลอดภัยเข้ามาเกี่ยวข้อง

1.4. วิธีการดำเนินงาน

การดำเนินงานในโครงการนี้ เริ่มจากการศึกษารายละเอียดของเว็บเซอร์วิสในส่วนต่างๆ ไม่ว่าจะเป็น XML, SOAP, WSDL, UDDI ฯลฯ จากนั้นจึงทำการเลือกชุดที่จะนำมาพัฒนาโดยในที่นี้ได้เลือกใช้ IBM Websphere Application Developer version 5.1.1 เนื่องจากมีความพร้อมในการสร้างเว็บเซอร์วิสมาก จากนั้นจึงได้ศึกษาวิธีการสร้างเว็บเซอร์วิสหลังจากนั้นจึงเริ่มศึกษาเกี่ยวกับ ความปลอดภัยของเว็บเซอร์วิส (WS-Security) และ การเชื่อมต่อระหว่างเว็บเซอร์วิส (WS-Interoperability) สุดท้ายจึงนำมาใช้ในเว็บเซอร์วิสระหว่าง Java Web Service – Java Web Service และ Java Web Service – ASP .Net Web Service

บทที่ 2

ทฤษฎีพื้นฐานเกี่ยวกับเว็บเซอร์วิส

2.1. เว็บเซอร์วิส

เว็บเซอร์วิส เป็นระบบซอฟต์แวร์ที่ออกแบบมาเพื่อสนับสนุนการทำงานระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ผ่านระบบเครือข่าย โดยที่ภาษาที่ใช้ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์คือภาษาเอ็กซ์เอ็มแอล (XML)

การอธิบายวิธีการใช้งานของเว็บเซอร์วิสนั้นจะอธิบายโดยใช้ภาษา WSDL (Web Services Description Language) ซึ่งเป็นภาษา XML ประเภทหนึ่ง

ระบบอื่นๆ จะสามารถติดต่อและทำงานกับเว็บเซอร์วิส โดยใช้ โปรโตคอลที่ชื่อว่า SOAP (Simple Object Protocol) ซึ่งใช้ภาษา XML เป็นมาตรฐานในการติดต่อระหว่างระบบโดยผ่านทางโปรโตคอลอื่นที่ใช้ในการส่งข้อมูลบนเว็บ อาทิเช่น โปรโตคอล HTTP

2.1.1. ประโยชน์ของการทำงานร่วมกันระหว่าง แอปพลิเคชัน กับ เว็บเซอร์วิส

เว็บเซอร์วิส ช่วยให้การเข้าถึงข้อมูลสารสนเทศจากแอปพลิเคชันที่ต่างกันเป็นไปได้โดยง่าย โดยแอปพลิเคชันนั้นๆ สามารถเขียนด้วย Java และรันอยู่บน Sun Solaris Application Server หรืออาจจะเขียนด้วย C++ และรันอยู่บน Windows NT หรืออาจจะเขียนด้วย Perl และรันอยู่บนเครื่อง Linux ซึ่งมาตรฐานของ เว็บเซอร์วิส ทำให้อินเทอร์เน็ตเฟสของแอปพลิเคชันเหล่านี้ ถูกอธิบายโดย WSDL และทำให้อยู่ในมาตรฐานของ UDDI หลังจากนั้น จึงสามารถติดต่อสื่อสารถึงกันโดย XML ผ่าน SOAP อินเทอร์เน็ต

เว็บเซอร์วิส สามารถถูกเรียกใช้ภายในองค์กรเองหรือจากภายนอกองค์กร โดยผ่านไฟร์วอลล์ ดังนั้นจึงมีองค์กรใหญ่ๆ มากมาย กำลังพัฒนาระบบที่มีอยู่ของตน ให้เข้ากับ เว็บเซอร์วิส ซึ่งนับเป็นการลงทุนที่คุ้มค่า เนื่องจาก เว็บเซอร์วิส สามารถเพิ่มศักยภาพในการทำงานขององค์กร อีกทั้งลดค่าใช้จ่ายในการจัดการทรัพยากรขององค์กรได้อีกทางหนึ่ง

นอกจากนั้น เว็บเซอร์วิส ยังสามารถใช้ร่วมกับ เว็บแอปพลิเคชัน โดยส่งผ่านข้อมูลทางอินเทอร์เน็ตได้อีกด้วยซึ่งนับเป็นวิธีที่มีประสิทธิภาพในการติดต่อสื่อสารกับลูกค้าหรือหุ้นส่วน ถึงแม้จะต้องคำนึงถึงระบบรักษาความปลอดภัย และการจัดการรายการของข้อมูลอยู่ก็ตาม แต่เว็บเซอร์วิสได้ใช้มาตรฐานทั่วไปของ Internet เรื่องดังกล่าวจึงนับเป็นเรื่องธรรมดาของการสื่อสารผ่านระบบอิเล็กทรอนิกส์

2.1.2. การทำงานของเว็บเซอร์วิส

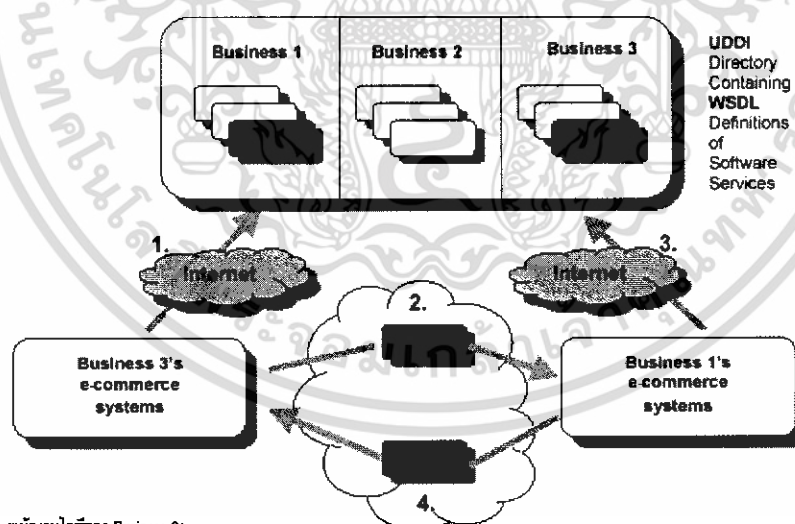
การทำงานของเว็บเซอร์วิสประกอบไปด้วยมาตรฐานหลัก 4 อย่างซึ่งอธิบายได้ง่ายๆ ดังนี้

1. **XML (Extensible Markup Language)** เป็นภาษามาตรฐานที่ทุกระบบสนับสนุนทำให้ข้อมูลที่มีโครงสร้างของภาษา XML จะถูกนำไปประมวลผลต่ออย่างอัตโนมัติได้อย่างง่ายดาย ภาษา XML จึงถูกนำมาใช้เป็นมาตรฐานในการแลกเปลี่ยนข้อมูลของ Web Services

2. **SOAP (Simple Object Access Protocol)** หรือโซฟ เป็นมาตรฐานของเทคโนโลยี Distributed Objects แบบหนึ่ง โดยทำหน้าที่ส่งข้อมูลผ่านอินเทอร์เน็ต ในรูปแบบของ XML ทำให้เรียกใช้งานโปรแกรมข้ามระบบผ่านทางอินเทอร์เน็ตได้

3. **WSDL (Web Services Description Language)** เป็นภาษามาตรฐานที่ใช้สำหรับอธิบายการใช้งานโปรแกรมที่เปิดให้บริการ ซึ่งเขียนขึ้นตามแบบมาตรฐาน XML ดังนั้น WSDL จึงเป็นเสมือนคู่มือให้กับระบบ เพื่อเรียนรู้วิธีการเรียกใช้งาน Web Services

4. **UDDI (Universal Description, Discovery, and Integration)** เป็นระบบมาตรฐานในการอธิบายและค้นหา Web Services โดยเป็นตัวกลางให้ Provider มาลงทะเบียนไว้ โดยใช้ไฟล์ WSDL บอกรายละเอียดของบริษัทและบริการที่มีให้ ทำให้ Requestor สามารถค้นหาและทราบว่าบริษัทมีผลิตภัณฑ์และบริการอะไรบ้าง สามารถติดต่อขอดำเนินธุรกิจการค้ากับบริษัทได้โดยอัตโนมัติผ่านทาง Web Services



พนักงานไอทีของ Business 3:

1. ค้นหาข้อกำหนดการให้บริการใน WSDL ผ่านอินเทอร์เน็ตของ Business 1
2. ติดต่อผ่านอินเทอร์เน็ตของซอฟต์แวร์ที่ให้บริการ (ด้วย SOAP) ผ่านทางอินเทอร์เน็ต
- 3,4. Business 1 รวมหรือบูรณาการกับ Business 3 ด้วยวิธีเดียวกัน

รูปที่ 2.1 อธิบายการทำงานของเว็บเซอร์วิส

จากมาตรฐานทั้ง 4 อย่างที่กล่าวข้างต้นสามารถสรุปลำดับขั้นตอนการทำงานของ Web Services ได้ดังนี้

1. Provider จัดทำระบบหรือบริการที่เป็น Web Services ขึ้นมา
2. ทำการลงทะเบียน Web Services กับหน่วยงานที่ให้บริการระบบ UDDI (หรือ Registry)
3. นำ WSDL ไฟล์ไปไว้ในระบบ UDDI ที่ได้ลงทะเบียนไว้
4. Requestor ทำการค้นหาระบบหรือบริการที่ต้องการจากระบบ UDDI
5. เมื่อ Requestor ได้พบระบบหรือบริการที่ต้องการจะนำไฟล์ WSDL ไปเรียนรู้วิธีการเรียก

ใช้ผ่านระบบของตน

6. Requestor ทำการติดต่อและเรียกใช้ระบบหรือบริการจาก Provider ได้โดยตรงผ่าน SOAP ในระบบของตน

2.1.3. เปรียบเทียบระหว่าง Web Application และ Web Services

เมื่อเข้าใจความหมายและการทำงานของ Web Application และ Web Services แล้ว จะเห็นว่าเครื่องมือทั้งสองต่างใช้ HTTP Protocol หรือ อินเทอร์เน็ต เป็นช่องทางในการสื่อสารเหมือนกัน แต่มีวัตถุประสงค์ต่างกัน โดย Web Application ใช้เพื่อการแลกเปลี่ยนไฟล์ HTML ระหว่าง เว็บเซิร์ฟเวอร์ แต่ Web Services เป็นการแลกเปลี่ยน “บริการ” (ก้อนโปรแกรม Software Components) ระหว่างระบบสารสนเทศผ่านเว็บเซิร์ฟเวอร์

ในเรื่องของความสามารถโดยส่วนใหญ่จะใช้ Web Application ในการติดต่อกับผู้ใช้ผ่านทาง Internet browser เพื่อนำเสนอข้อมูลและการทำงานต่างๆ ส่วน Web Services จะทำหน้าที่ในการติดต่อกับ เว็บเซิร์ฟเวอร์ เพื่อแลกเปลี่ยนข้อมูลและการทำงานหรือให้บริการข้ามระบบกันโดยใช้ Web Application หรือ Application Interface ในการติดต่อกับผู้ใช้ นอกจากนี้ Web Services ยังสามารถทำงานกับระบบต่างๆ ได้มากกว่า 1 ระบบ ในขณะที่ Web Application ไม่สามารถทำได้โดยตรง ซึ่งสามารถสรุปการเปรียบเทียบได้ดังตาราง

ตารางที่ 2.1 แสดงการเปรียบเทียบระหว่าง เว็บแอปพลิเคชันกับเว็บเซอร์วิส

หัวข้อ	Web Services	Web Applications
การเชื่อมต่อ	program-program	human-program
ภาษาที่ใช้	XML	HTML
รายชื่อการให้บริการ	ค้นหาผ่าน UDDI	ค้นหาผ่าน search engine
ขอบเขตการใช้งาน	Business-to-Business (B2B)	Business-to-Customer (B2C)
โปรโตคอล(Protocol)	SOAP+HTTP	HTTP

หลังจากที่ได้ทำความเข้าใจความหมายและข้อเปรียบเทียบระหว่าง Web Application และ Web Services แล้ว ความแตกต่างระหว่างเครื่องมือทั้งสองจะเห็นได้ชัดในเรื่องของความสามารถ แต่ในการนำไปใช้งานจำเป็นจะต้องประยุกต์ใช้เครื่องมือทั้งสองชนิดร่วมกัน เพื่อให้เกิดประสิทธิภาพมากที่สุด

2.1.4. การประยุกต์ใช้งานเว็บเซอร์วิส

ขณะเดียวกัน เว็บเซอร์วิส เป็นพื้นฐานสำคัญของการบริการใหม่ๆ ในลักษณะที่ต้องการเชื่อมโยง ระบบสารสนเทศที่มีความแตกต่างเข้าด้วยกันอย่างอัตโนมัติ เช่น การพาณิชย์อิเล็กทรอนิกส์ ระหว่างธุรกิจกับธุรกิจ (B2B), ระบบการบริการรัฐบาลอิเล็กทรอนิกส์ ในลักษณะที่มีการเชื่อมบริการของภาครัฐกับภาครัฐเข้าด้วยกัน เพื่อทำให้เกิดการบริการต่อประชาชนจากจุดเดียวได้ (One Stop Services) เป็นต้น

ตัวอย่างพื้นฐานหนึ่ง คือ การเชื่อมโยงระหว่างระบบการสั่งซื้อ ของบริษัทผู้ขายบริษัทหนึ่ง ไปยังระบบสารสนเทศการ จัดส่งของบริษัทจัดส่งอีกบริษัทหนึ่ง จะเป็นไปได้ง่ายมากขึ้นด้วยเทคโนโลยีของเว็บเซอร์วิส พื้นฐานดังกล่าวนี้ จะทำให้ "การจัดการห่วงโซ่อุปทานด้วยสื่ออิเล็กทรอนิกส์" (e-SCM: Supply Chain Management) และธุรกิจผ่านสื่ออิเล็กทรอนิกส์ที่มีประสิทธิภาพ และมีความคุ้มค่ามากขึ้น โดยมีการแลกเปลี่ยนข้อมูลธุรกิจ ระหว่างองค์กรในห่วงโซ่อุปทาน

ทั้งนี้แต่ละบริษัทจะมีระบบงาน (แอปพลิเคชัน) และใช้รูปแบบของข้อมูล (ฟอร์แมต คำว่า) ที่แตกต่างกันได้ ซึ่งเมื่อมีการส่งข้อมูลจากองค์กรหนึ่งไปอีกองค์กรหนึ่ง ระบบคอมพิวเตอร์ของบริษัทผู้รับ สามารถนำข้อมูล ที่ได้ไปประมวลผลต่อได้ทันที การเชื่อมโยงระบบอีคอมเมิร์ซของหน้าร้าน กับระบบหลังร้าน แล้วยังเชื่อมโยง กับระบบสารสนเทศของพันธมิตรทางการค้านี้ จะทำให้การทำธุรกิจผ่านสื่ออิเล็กทรอนิกส์ มีความคุ้มค่า มีประสิทธิภาพ และสร้างผลกำไรให้กับบริษัทได้

2.2. Service-Oriented Architecture (SOA)

แนวคิดหลักของธุรกิจผ่านสื่ออิเล็กทรอนิกส์ คือกลยุทธ์การเพิ่มประสิทธิภาพของการดำเนินธุรกิจโดยใช้ อินเทอร์เน็ต อินทราเน็ตและเอ็กซ์ทราเน็ตในทุกระดับอย่างต่อเนื่อง

ปัจจุบันการพัฒนาระบบธุรกิจผ่านสื่ออิเล็กทรอนิกส์อาจจะใช้สถาปัตยกรรมการบริการในลักษณะที่เรียกว่า "Service- Oriented Architecture" (SOA) เป็นแนวคิดเบื้องต้น ระบบและแอปพลิเคชันส่วนใหญ่ในโลกของธุรกิจที่ใช้งานในปัจจุบันเป็นApplication

SOA มีส่วนประกอบหลักสามส่วนคือ

- ผู้ให้บริการ (Service provider)
- ผู้ขอบริการ (Service requester)
- ตัวแทนของผู้ให้บริการ (Service broker)

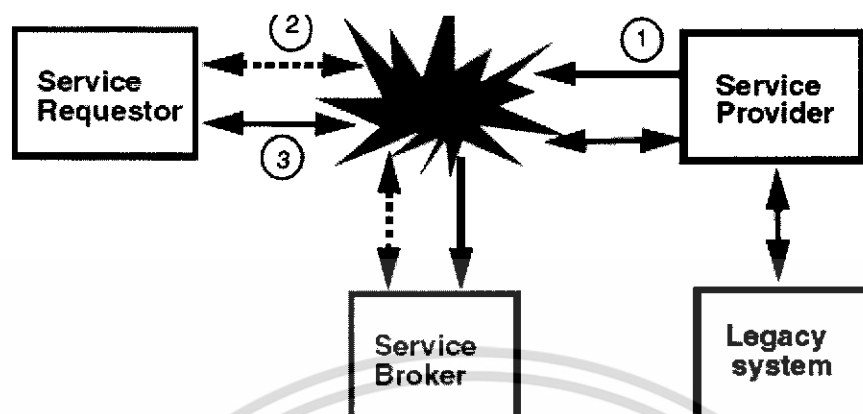


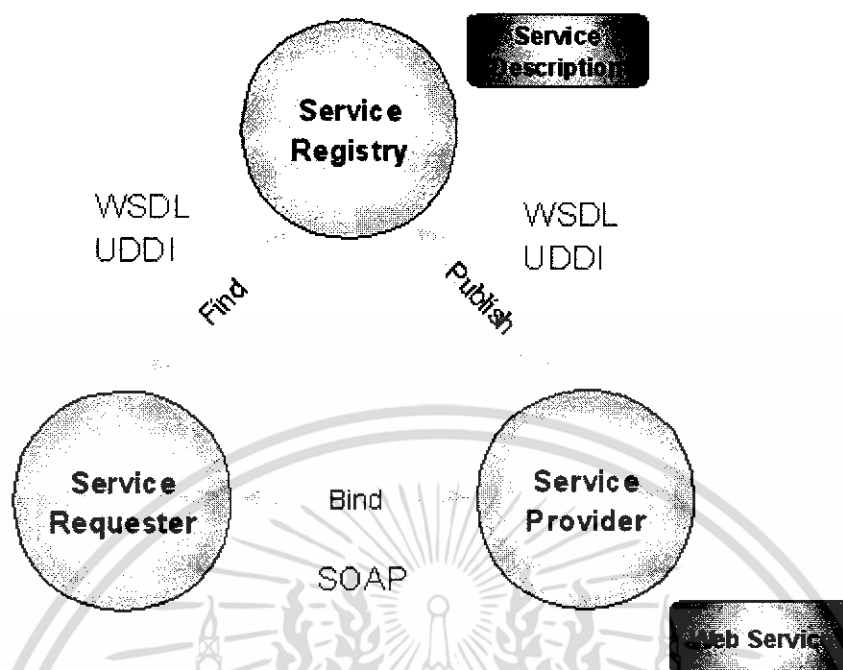
Figure 1-1 Web services roles and operations

รูปที่ 2.2 แสดงส่วนประกอบและการทำงานของ SOA

ซึ่งส่วนประกอบหลักทั้ง 3 ส่วนนี้ติดต่อถึงกันโดยใช้ฟังก์ชันพื้นฐาน คือ

- การประกาศ (publish)
- การค้นหา (find)
- การเรียกใช้ (bind)

ฟังก์ชันทั้งสามมีการทำงานดังนี้คือ ผู้ให้บริการ (Service provider) ทำการประกาศ (publish) บริการไปยังตัวแทนของผู้ให้บริการ (Service broker) หรือที่อาจเรียกว่า "ไดเรกทอรีของบริการ" ในขณะที่ผู้ขอบริการ (Service requester) จะทำการค้นหา (find) บริการที่ต้องการ และเมื่อพบเห็นก็จะทำการเรียกใช้ (bind) ไป



รูปที่ 2.3 แสดงการติดต่อกันของส่วนประกอบหลัก

- ข้อดี SOA

เนื่องจากสถาปัตยกรรม SOA ของเว็บเซอร์วิสทำให้สามารถนำเอาโค้ดที่เขียนไปแล้วกลับมาใช้ใหม่ซึ่งไม่เพียงทำให้ประหยัดเวลาในการพัฒนาแต่ยังช่วยประหยัดต้นทุนส่วนเพิ่มทั้งในเรื่องฮาร์ดแวร์และซอฟต์แวร์ได้อีกด้วย

2.3. XML

XML (The Extensible Markup Language 1.0) เป็นภาษา Markup ที่เป็น text-based ซึ่งทำให้เป็น มาตรฐานในการแลกเปลี่ยนข้อมูลบนอินเทอร์เน็ตอย่างรวดเร็ว ผู้ที่ทำหน้าที่รับผิดชอบ และกำหนดมาตรฐานของ XML คือ World Wide Web Consortium (W3C) ความแตกต่างระหว่าง XML กับ HTML คือ HTML ถูกนำมาใช้ในการสร้าง เว็บเพจ ที่สามารถแสดงผลได้โดยโปรแกรมเบราว์เซอร์ แต่ XML จะใส่ tags ได้อย่างอิสระ แล้วทำการส่ง XML ชุดนี้ไป ประมวลผลยังแอปพลิเคชันใด ๆ ที่สามารถใช้ข้อมูลใน XML นี้

2.3.1. XML & HTML

HTML ภาษาที่ใช้ในการเขียน Web มากที่สุดนั่นเป็นเพราะมีรูปแบบที่ง่ายต่อการแสดงผลของ Browser เนื่องจาก มี tag คายตัวที่สามารถบอกได้ว่าเมื่อเจอ tag นี้จะแสดงผลอย่างไร เช่น เมื่อเจอ tag `<B>...</B>` ในเอกสารก็ให้แสดงข้อความที่อยู่ระหว่าง tag เป็นตัวหนา แต่จะสังเกตได้ว่าคอมพิวเตอร์จะไม่เข้าใจว่าข้อความนั้นคืออะไร เพียงแต่รู้ว่าแสดงผลอย่างไร นั่นแสดงว่าไม่สามารถนำข้อมูลภายใน tag เหล่านี้ไปทำการประมวลใดๆ ได้เลย

XML เป็นภาษาที่มีลักษณะเป็น tag คล้าย HTML แต่ไม่ได้มุ่งที่การแสดงผล XML มุ่งที่การสื่อความหมายโดยอนุญาตให้ผู้ใช้งานสามารถกำหนด tag ขึ้นได้เองเพื่อให้สื่อความหมายทางภาษาของมนุษย์ แต่คอมพิวเตอร์เองก็เข้าใจเช่นกัน ทำให้ข้อมูลระหว่าง tag สามารถนำไปประมวลผลต่อได้ เช่น

```
<ComputerBook>
  <book>
    <name>เว็บเซอร์วิส</name>
    <price>10.00$</price>
  </book>
  <book>
    <name>xml</name>
    <price>10.00$</price>
  </book>
</ComputerBook>
```

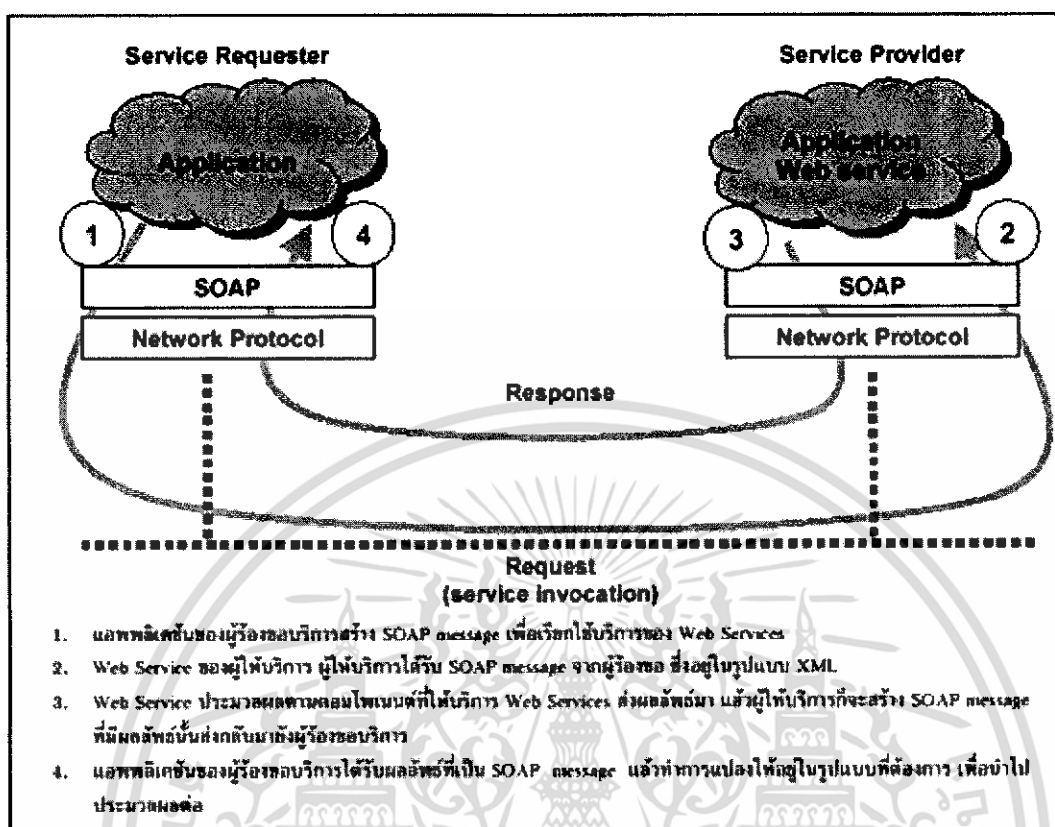
รูปที่ 2.4 แสดง Tag ของภาษา XML

จากตัวอย่าง จะเห็นได้ว่ารูปแบบเอกสาร xml ไม่ได้บอกวิธีแสดงผลไว้ แต่เอกสารสามารถสื่อความหมายให้คอมพิวเตอร์เข้าใจได้ และนำค่าไปประมวลผลต่อได้ แต่ถ้าเราต้องการจะแสดงผลเราต้องใช้ควบคู่กันกับ HTML

2.4. SOAP (Simple Object Access Protocol)

SOAP กลายเป็นสิ่งที่มีความสำคัญสำหรับ Web Services อย่างรวดเร็ว เป็นโปรโตคอลที่ผู้จัดหา Web Services เลือกใช้ที่จะส่ง message ระหว่าง Web Services SOAP เป็น Transport Protocol ที่มี XML เป็นพื้นฐานและใช้ HTTP เป็นโปรโตคอลร่วมในการส่งผ่านเครือข่าย SOAP จะระบุวิธีการเข้ารหัสส่วนหัว (Header Encoding) ของทั้ง HTTP และไฟล์ XML ให้อย่างชัดเจนทั้งใน ส่วนของการติดต่อไปยังคอมพิวเตอร์อีกเครื่องหนึ่งและส่งผ่านข้อมูลไปให้ รวมถึงระบุวิธีที่โปรแกรมซึ่งถูกเรียกนั้นจะส่งค่าคืนกลับมาด้วย SOAP (Simple Object Access Protocol) เป็น XML-based โปรโตคอล (lightweight protocol) และใช้ HTTP เป็นโปรโตคอลร่วม สำหรับการแลกเปลี่ยนข้อมูลในสภาวะแวดล้อมแบบกระจายศูนย์ (decentralized, distributed environment)

SOAP ได้ กำหนดเมสเสจิงโปรโตคอล (Messaging Protocol) ระหว่างผู้ขอบริการ (requestor) กับผู้ให้บริการ (provider) เช่น ผู้ขอบริการสามารถติดต่อแลกเปลี่ยนข้อมูลกับผู้ให้บริการโดยใช้ RMI (Remote Method Invocation) ตามวิธีการของ โปรแกรมแบบออปเจ็ค บริษัทไมโครซอฟท์, ไอบีเอ็ม, โลตัส, ยูสเซอร์แลนด์ (UserLand) และ ดีเวลลอปเปอร์เมนเตอร์ (DeveloperMenter) ได้ร่วมกันกำหนดมาตรฐานของ SOAP ขึ้น ซึ่งต่อมาได้มีบริษัทอีก 30 กว่าบริษัทเข้าร่วมและ จัดตั้งเป็น W3C XML Protocol Workgroup ขึ้น SOAP ได้กำหนดรูปแบบพื้นฐานของการสื่อสารแบบกระจายขึ้น โดยการ พัฒนา SOA แม้ว่า SOA จะไม่ได้กำหนดเมสเสจิงโปรโตคอล (Messaging Protocol) ไว้ แต่ SOAP ได้ถูกกำหนด ให้เป็น Services-Oriented Architecture Protocol เรียบร้อยแล้ว เนื่องจากมัน ได้ถูกใช้ในการ พัฒนา SOA อย่างแพร่หลายแล้วนั่นเอง จุดเด่นของ SOAP ก็คือเป็นโปรโตคอลที่เป็นกลาง กล่าวคือ ไม่มีใครเป็นเจ้าของและเป็นโปรโตคอล ที่ทำงานกับโปรโตคอลอื่นหลายชนิด การพัฒนาก่อนุญาตให้ ทำได้อย่างอิสระตามแพลตฟอร์มระบบปฏิบัติการ แบบจำลองทางวัตถุ (Object model) และภาษา โปรแกรมของผู้ที่ทำการพัฒนา



รูปที่ 2.5 แสดง XML messaging using SOAP

2.4.1. ทำไมต้องเป็น SOAP

เนื่องจากจุดประสงค์หลักของการใช้งานเว็บเซอร์วิส เราต้องการให้แอปพลิเคชัน มีการทำงานกับแอปพลิเคชันที่ทำงานอยู่ในเครื่องอื่นโดยผ่านทางเครือข่าย ซึ่งเทคโนโลยีที่มีอยู่ในปัจจุบันที่ใช้มีการสื่อสารระหว่าง Object ในระยะไกล (Remote Procedure Calls : RPC) เช่น DCOM,EJB หรือ CORBA นั้นไม่ได้ถูกออกแบบมาใช้สำหรับโปรโตคอล HTTP (เครือข่ายที่ปัจจุบันใช้งานแพร่หลายและสะดวกมากที่สุดได้แก่เครือข่าย Internet ซึ่งให้การสื่อสารผ่านโปรโตคอล HTTP เป็นส่วนใหญ่)

เทคนิค RPC ของเทคโนโลยีที่กล่าวข้างต้นนั้นต่างก็มีปัญหาในด้านการนำมาใช้งานในแง่ของความเข้ากันได้ของการเรียกใช้งานข้ามเทคโนโลยี เนื่องจากเป็นเทคโนโลยีเฉพาะของแต่ละค่าย ยกเว้น CORBA ผู้พัฒนาระบบจะต้องพัฒนาโปรแกรมที่มีความซับซ้อน และยังมีปัญหาในส่วนของ Firewalls และ Proxy Server ด้วยเนื่องจากโดยปกติ เซอร์เวอร์จะปิดการสื่อสารที่ไม่ใช่โปรโตคอล HTTP ออกไปเพื่อความปลอดภัยของระบบที่มีการติดต่อสื่อสารกับภายนอก

ดังนั้นทางเลือกของการสื่อสารที่จะนำมาใช้ในการให้บริการเว็บเซอร์วิส ก็คือให้ทำงานอยู่บนโปรโตคอล HTTP ไปเลย ซึ่ง SOAP นอกจากจะทำงานบนโปรโตคอล HTTP แล้วยังเป็นมาตรฐานเปิด

ที่จะทำให้สามารถติดต่อสื่อสารกับเครื่องคอมพิวเตอร์ที่มีความแตกต่างกันทั้งระบบปฏิบัติการ, เทคโนโลยี รวมไปถึงภาษาที่ใช้ในการพัฒนาด้วยก็ได้

2.4.2. โครงสร้างของ SOAP

เอกสาร SOAP นั้นมีโครงสร้างในรูปแบบ XML ซึ่งเราสามารถแบ่งเป็นส่วนของเอกสารได้เป็น 3 ส่วนหลักดังนี้คือ

1. SOAP envelop เนื้อหาสาระ (Content) ของเอกสารทั้งหมด
2. SOAP header ส่วนเพิ่มเติมของเอกสาร SOAP ซึ่งจะมีก็ได้ หรือ ไม่มีก็ได้
3. SOAP body ส่วนที่ใช้ในการเรียกใช้งานเซอร์วิส และผลลัพธ์ที่ได้จากเซอร์วิส



รูปที่ 2.6 โครงสร้างของเอกสาร SOAP

```
<soap:Envelope>
  <soap:Body>
    <GetPrice>
      <Item>Rose</Item>
      <Quantity>100</Quantity>
    </GetPrice>
  </soap:Body>
</soap:Envelope>
```

รูปที่ 2.7 ตัวอย่างเอกสาร SOAP อย่างง่ายของการสอบถามราคา ดอกกุหลาบ จำนวน 100 ดอก

อีกตัวอย่างหนึ่งของเอกสาร SOAP ที่จะแสดงในรูปด้านล่างเพื่อจะทำความเข้าใจในส่วนต่างๆ ที่มีความซับซ้อนขึ้น แต่ก็อย่าจนเกินไปที่จะทำความเข้าใจได้ดัง รูปที่ 2.8 SOAP Envelope ต่อไปนี้

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Header>
    <!-- ข้อมูลในส่วนของ Header -->
    <i:local xmlns:i="http://www.i3t.or.th/ws/">
      <i:currency>Bath</i:currency>
    </i:local>
  </soap:Header>
  <soap:Body>
    <!-- ข้อมูลในส่วนของ Body -->
    <GetPrice>
      <Item>Rose</Item>
      <Quantity>100</Quantity>
    </GetPrice>
  </soap:Body>
  <soap:Fault>
    <!-- ข้อมูลของ SOAP ในกรณีมีข้อผิดพลาด จาก SOAP Node -->
  </ soap:Fault>
</soap:Envelope>
```

รูปที่ 2.8 ตัวอย่างเอกสาร SOAP

ตารางที่ 2.2 แสดงส่วนประกอบของ SOAP

Element	Definition
<Envelope>	จะเป็น Root ของเอกสาร XML SOAP เสมอ โดยจากตัวอย่างจะใช้ Namespaces จาก "http://schemas.xmlsoap.org/soap/envelope/" (Schema for the SOAP/1.1 envelope) ดังนั้น Element ต่างๆ ในตัวอย่างนี้ที่อยู่ในเอกสาร XML ส่วนที่เป็นมาตรฐานของ SOAP เราจะมี Namespaces ที่เรากำลังเป็น soap (<soap:Envelop>, <soap:Header>, <soap:Body>, <soap:Fault>) Attribute ใน Envelop Attribute encodingStyle ในตัวอย่างมีการอ้างการ encoding จาก "http://schemas.xmlsoap.org/soap/encoding/" (Schema for the SOAP/1.1 encoding) จะใช้ในกรณีที่เรามีการใช้ Parameter หลายชนิดข้อมูลในเอกสาร (Data Type) เช่น Boolean, String, Integer เป็นต้น
<Header>	จากตัวอย่างมีการเพิ่มเนื้อหาของเอกสาร SOAP เข้าไปในส่วนของ Header ในที่นี้คือ สกุลเงิน (Currency) ซึ่งส่วนนี้จะใช้ในส่วนของการ Application ไม่ได้เป็นส่วนหนึ่งของมาตรฐานของ SOAP แต่ผู้ใช้งานเป็นผู้กำหนดใช้เอง (User-defined) Attribute ใน Header ใน <Header> Element อาจจะมีการใส่ Attribute mustUnderstand เพื่อให้ฝั่งที่รับเอกสาร SOAP ให้ทำการประมวลผลในส่วน Header ด้วยโดยกำหนดค่าเป็น Boolean เช่น <pre> <i:local xmlns:i="http://www.i3t.or.th/ws/"> <i:country mustUnderstand="1">Thailand</i:country> <i:currency mustUnderstand="0">Bath</i:currency> </i:local> </pre> ค่าของ mustUnderstand ถ้าเป็น "0" แสดงว่าไม่ต้องประมวลผล ถ้าเป็น "1" : true คือจำเป็นต้องมีการประมวลผล (โดยค่า default = "0" : fault)

<Body>	โดยปกติแล้วเอกสาร SOAP จะต้องมีส่วนของ Body เพราะเป็นส่วนเป็นเนื้อหาสาระจริงๆ ของ SOAP จากตัวอย่างเราต้องการสอบถามราคาของดอกกุหลาบจำนวน 100 ดอก ซึ่ง <GetPrice>, <Item> และ <Quantity> เป็น Element ที่ใช้งานใน Application ไม่ใช่มาตรฐานของ SOAP
<Fault>	เป็นส่วนที่จะถูกใช้เมื่อมีข้อผิดพลาดในการประมวลผลของเอกสาร SOAP ซึ่งโดยปกติจะเห็นเฉพาะที่เป็นเอกสารตอบกลับเท่านั้น (Reply Message หรือ Answer Message) ซึ่งตัวอย่างจะเป็นเอกสารที่เป็นการร้องขอบริการ (Request Message) ซึ่งไม่น่าจะมีส่วนของ <Fault> Element แต่ที่เขียนไว้เพื่อให้เห็นภาพรวมของโครงสร้างเอกสาร SOAP ทั้งหมดก่อน

```

<soap:Fault>
  <faultcode>Client</faultcode>
  <faultstring>Invalid Request</faultstring>
</soap:Fault>

```

รูปที่ 2.9 ตัวอย่างข้อมูลใน <Fault> Element ที่มีการเกิดข้อผิดพลาด

ตารางที่ 2.3 อธิบาย Fault Element

Sub Element	คำอธิบาย
<faultcode>	ชื่อของการเกิดข้อผิดพลาด (ดูตารางด้านล่าง)
<faultstring>	คำอธิบายของข้อผิดพลาด
<faultactor>	ชื่อของแหล่งที่เกิดข้อผิดพลาด
<detail>	ข้อมูลรายละเอียดเพิ่มเติมของข้อผิดพลาดนั้น

ตารางที่ 2.4 แสดง Fault code Error

Error (faultcode)	คำอธิบาย
VersionMismatch	มีข้อผิดพลาดของการอ้าง Namespace ของ Element
MustUnderstand	ในการประมวลผลของ Sub Element ใน Header ที่มีการกำหนด attribute mustUnderstand เป็น "1" ไม่สามารถประมวลผลได้ (not understood)
Client	เอกสาร SOAP มีข้อผิดพลาดอันเนื่องมาจากมีโครงสร้างทางผิด หรือมีข้อมูลที่ไม่ถูกต้อง
Server	มีข้อผิดพลาดจาก Server ไม่สามารถประมวลผลเอกสารได้

จะเห็นว่า SOAP Envelope ก็เป็นเอกสารที่อยู่ในรูปแบบ XML ทั่วไป แต่สิ่งที่ทำให้ SOAP มีความสามารถมากขึ้นก็คือ SOAP ได้มีการกำหนดโครงสร้างของเอกสารเป็นส่วนๆ โดยอธิบายว่า ส่วนใดมีหน้าที่อะไร (What) มีข้อมูลอะไรอยู่ในส่วนนั้น และใคร (Who) คือผู้ที่ต้องสนใจในส่วนนั้นๆ และกล่าวถึงว่าส่วนใดจำเป็นที่ต้องมีในเอกสาร และส่วนใดเป็นส่วนที่เพิ่มเติมอาจจะมีหรือไม่มี ส่วนนั้นก็ได้ขึ้นอยู่กับความจำเป็นในการใช้งาน ทำให้การพัฒนาโปรแกรมมีทิศทางที่ค่อนข้างชัดเจน แต่สามารถเพิ่มขยายได้ในอนาคตอีกด้วย

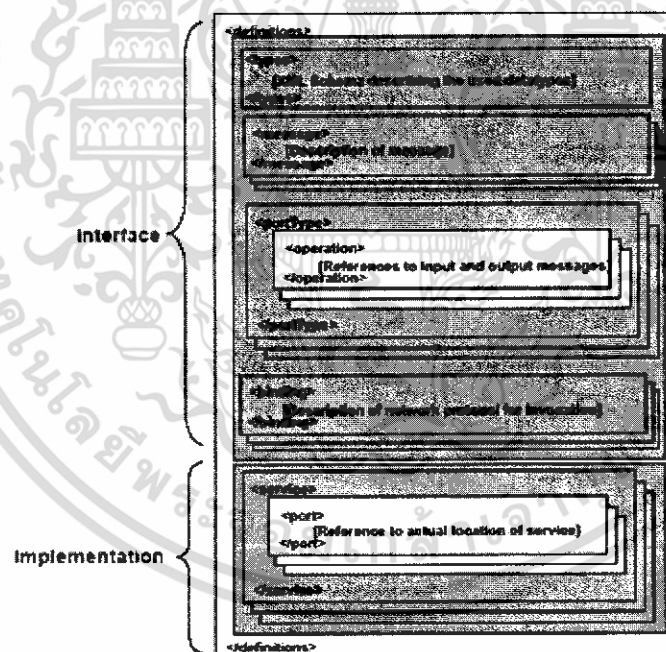
2.5. WSDL (Web Services Description Language)

WSDL (Web Services Description Language) เป็นภาษาที่ใช้อธิบายคุณลักษณะการให้บริการของ Web Services และวิธีการติดต่อกับ Web Services ความต้องการของนิยามนี้เกี่ยวเนื่องกับความต้องการของ distributed system ที่จะกำหนด Interface Definition Language (IDL) โดยใช้ภาษา XML

WSDL คือ มาตรฐานสำหรับการประกาศ process ที่จำเป็นในการเรียกใช้เซอร์วิส SOAP (Simple Object Access Potocal)

2.5.1. โครงสร้างเอกสาร WSDL

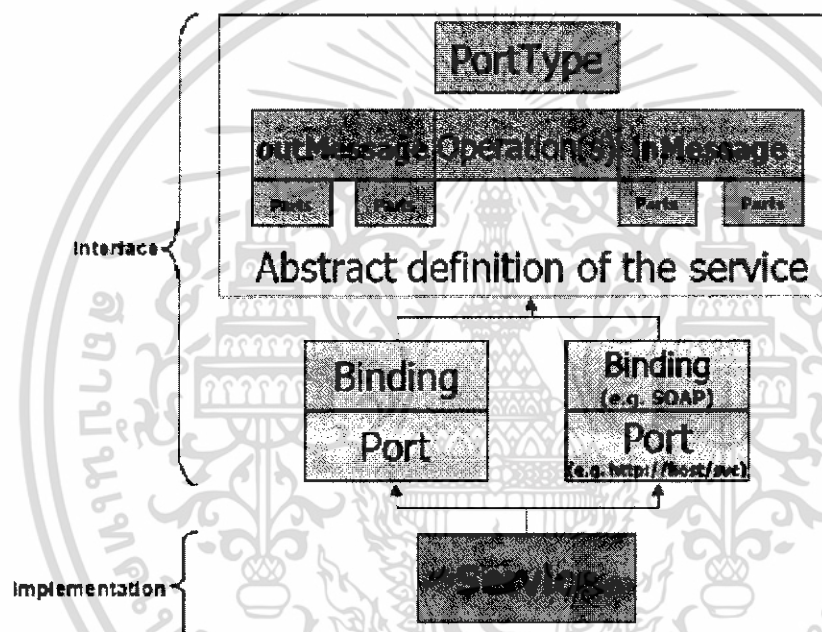
WSDL เป็นภาษาที่อยู่ในความดูแลขององค์กร W3C (World Wide Web Consortium) version ที่มีอยู่ในปัจจุบัน คือ WSDL 1.1 ในการใช้งานจริง หากเราสร้างบริการ Web Services ก็จะมีเครื่องมือช่วยสร้างเอกสาร WSDL สำหรับ Web Services อย่างอัตโนมัติ



รูปที่ 2.10 แสดง ส่วนประกอบของ WSDL ในรูปแบบ XML

WSDL(Web Services Description Language)ใช้กำหนดเว็บเซอร์วิสหรือ อธิบายในรูปแบบพื้นฐานของ XML schema ไว้สำหรับ

- interfaces
- operation type
- ข้อมูลที่ถูกใช้ สำหรับ เว็บเซอร์วิส
- Data type(XML schema)
- Access protocol เช่น SOAP บน HTTP
- Contact endpoints เช่น บริการบน URL(s)



รูปที่ 2.11 แสดง ส่วนประกอบ WSDL ในรูปแบบโครงสร้าง

ส่วนประกอบของ WSDL

- แต่ละ message จะเป็นลำดับรายการในส่วนของ Parts
- Part ง่ายหรือซับซ้อนของ data type ขึ้นอยู่กับการกำหนดที่ดี
- Operation กำหนดระหว่าง การจับคู่ของ input & output ของข้อมูล
- แต่ละ Interface (portType) เป็นกลุ่มของ related operations
- Binding เกี่ยวข้องกับ interface (portType) ด้วยข้อมูลของ transport protocol
- แต่ละ port มีความเกี่ยวข้องกับ binding
- Service อาจจะใช้ได้ ที่หนึ่งหรือมากกว่า endpoint (ports)

จุดภายในเอกสารที่เราควรรู้เกี่ยวกับการติดต่อและเรียกใช้บริการของ Web Services มีจุดที่ควรรู้ ดังนี้

ตารางที่ 2.5 แสดง WSDL ELEMENT

Element	Definition
<types> ①	อธิบายชนิดข้อมูลที่ web service ใช้ เพื่อความเป็นกลาง WSDL ใช้ XML Schema syntax ในการระบุชนิดข้อมูล
<message> ②	อธิบาย data elements ของ operation แต่ละ message อาจมีมากกว่าหนึ่งส่วนเทียบได้กับ parameter ของ function ในการเขียนโปรแกรม
<portType> ③	เป็นส่วนที่สำคัญที่สุดใน WSDL element อธิบาย operations ที่ web service มีให้บริการและ messages ที่เกี่ยวข้อง เทียบได้กับ function library หรือ module หรือ class ในการเขียนโปรแกรม
<binding> ④	อธิบาย format ของ message และ protocol details ในแต่ละ port
<service> ⑤	สำหรับ web server จะมี Web Services จำนวนกี่บริการก็ได้ และ ชื่อ Web Services ก็เป็นตัวจำแนกและบ่งบอกแต่ละบริการซึ่งห้ามมีชื่อซ้ำกัน
<operation> ⑥	อธิบาย method ที่ให้บริการ Web Services หนึ่งจะมี method จำนวนกี่ method ก็ได้

ตามทฤษฎีแล้ว ไฟล์เอกสาร WSDL แต่ละไฟล์ สามารถอธิบายคุณลักษณะของบริการ Web Services ได้มากกว่า 1 บริการ โดยแต่ละ Web Services จะมี port สื่อสารเฉพาะตัว ซึ่งบ่งบอกไว้ในเอกสาร WSDL อยู่แล้ว

2.5.2. WSDL Example

```
<wsdl:definitions targetNamespace="http://WS" xmlns:impl="http://WS"
xmlns:intf="http://WS" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```

<wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://WS"
    xmlns="http://www.w3.org/2001/XMLSchema" xmlns:impl="http://WS"
    xmlns:int="http://WS" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <element name="getCourseAll">
      <complexType>
        <sequence>
        </complexType>
      </element>
    <element name="getCourseAllResponse">
      <complexType>
        <sequence>
          <element name="getCourseAllReturn" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
    <element name="buyClass">
      <complexType>
        <sequence>
          <element name="ClassID" type="xsd:int"/>
          <element name="Email" nillable="true" type="xsd:string"/>
          <element name="ReceiveID" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
    <element name="buyClassResponse">
      <complexType>
        <sequence>
          <element name="buyClassReturn" type="xsd:boolean"/>
        </sequence>
      </complexType>
    </element>
    <element name="reservation">
      <complexType>
        <sequence>
          <element name="ClassID" type="xsd:int"/>
          <element name="Email" nillable="true" type="xsd:string"/>
          <element name="Name" nillable="true" type="xsd:string"/>
          <element name="Surname" nillable="true" type="xsd:string"/>
          <element name="PhoneNo" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
    <element name="reservationResponse">
      <complexType>

```

```

        <sequence>
            <element name="reservationReturn" type="xsd:boolean"/>
        </sequence>
    </complexType>
</element>
<element name="getCourse">
    <complexType>
        <sequence>
            <element name="Subject" nillable="true" type="xsd:string"/>
            <element name="Period" type="xsd:int"/>
        </sequence>
    </complexType>
</element>
<element name="getCourseResponse">
    <complexType>
        <sequence>
            <element name="getCourseReturn" nillable="true" type="xsd:string"/>
        </sequence>
    </complexType>
</element>
<element name="getCourseFromEmail">
    <complexType>
        <sequence>
            <element name="Email" nillable="true" type="xsd:string"/>
        </sequence>
    </complexType>
</element>
<element name="getCourseFromEmailResponse">
    <complexType>
        <sequence>
            <element name="getCourseFromEmailReturn" nillable="true"
                type="xsd:string"/>
        </sequence>
    </complexType>
</element>
<element name="cancel">
    <complexType>
        <sequence>
            <element name="ClassID" type="xsd:int"/>
            <element name="Email" nillable="true" type="xsd:string"/>
        </sequence>
    </complexType>
</element>
<element name="cancelResponse">
    <complexType>
        <sequence>

```

```

        <element name="cancelReturn" type="xsd:boolean"/>
      </sequence>
    </complexType>
  </element>
</schema>
</wsdl:types>

```

```

<wsdl:message name="getCourseFromEmailRequest">
  <wsdl:part element="intf:getCourseFromEmail" name="parameters"/>
</wsdl:message>
<wsdl:message name="buyClassResponse">
  <wsdl:part element="intf:buyClassResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="getCourseFromEmailResponse">
  <wsdl:part element="intf:getCourseFromEmailResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="getCourseAllResponse">
  <wsdl:part element="intf:getCourseAllResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="getCourseAllRequest">
  <wsdl:part element="intf:getCourseAll" name="parameters"/>
</wsdl:message>
<wsdl:message name="cancelRequest">
  <wsdl:part element="intf:cancel" name="parameters"/>
</wsdl:message>
<wsdl:message name="reservationRequest">
  <wsdl:part element="intf:reservation" name="parameters"/>
</wsdl:message>
<wsdl:message name="getCourseRequest">
  <wsdl:part element="intf:getCourse" name="parameters"/>
</wsdl:message>
<wsdl:message name="getCourseResponse">
  <wsdl:part element="intf:getCourseResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="buyClassRequest">
  <wsdl:part element="intf:buyClass" name="parameters"/>
</wsdl:message>
<wsdl:message name="reservationResponse">
  <wsdl:part element="intf:reservationResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="cancelResponse">
  <wsdl:part element="intf:cancelResponse" name="parameters"/>
</wsdl:message>

```

2

```

<wsdl:portType name="ApplyPhysic">
  <wsdl:operation name="getCourseAll">

```

3

```

    <wsdl:input message="intf:getCourseAllRequest" name="getCourseAllRequest"/>
    <wsdl:output message="intf:getCourseAllResponse" name="getCourseAllResponse"/>
  </wsdl:operation>
  <wsdl:operation name="buyClass">
    <wsdl:input message="intf:buyClassRequest" name="buyClassRequest"/>
    <wsdl:output message="intf:buyClassResponse" name="buyClassResponse"/>
  </wsdl:operation>
  <wsdl:operation name="reservation">
    <wsdl:input message="intf:reservationRequest" name="reservationRequest"/>
    <wsdl:output message="intf:reservationResponse" name="reservationResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getCourse">
    <wsdl:input message="intf:getCourseRequest" name="getCourseRequest"/>
    <wsdl:output message="intf:getCourseResponse" name="getCourseResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getCourseFromEmail">
    <wsdl:input message="intf:getCourseFromEmailRequest"
      name="getCourseFromEmailRequest"/>
    <wsdl:output message="intf:getCourseFromEmailResponse"
      name="getCourseFromEmailResponse"/>
  </wsdl:operation>
  <wsdl:operation name="cancel">
    <wsdl:input message="intf:cancelRequest" name="cancelRequest"/>
    <wsdl:output message="intf:cancelResponse" name="cancelResponse"/>
  </wsdl:operation>
</wsdl:portType>

```

6

```

<wsdl:binding name="Apply" type="tns:SoapBinding" type="tns:ApplyPhysic">
  <wsdl:soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="getCourseAll">
    <wsdl:soap:operation soapAction=""/>
    <wsdl:input name="getCourseAllRequest">
      <wsdl:soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="getCourseAllResponse">
      <wsdl:soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="buyClass">
    <wsdl:soap:operation soapAction=""/>
    <wsdl:input name="buyClassRequest">
      <wsdl:soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="buyClassResponse">
      <wsdl:soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>

```

4

```

</wsdl:operation>
<wsdl:operation name="reservation">
  <wsdl:soap:operation soapAction="" />
  <wsdl:input name="reservationRequest">
    <wsdl:soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="reservationResponse">
    <wsdl:soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getCourse">
  <wsdl:soap:operation soapAction="" />
  <wsdl:input name="getCourseRequest">
    <wsdl:soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="getCourseResponse">
    <wsdl:soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="getCourseFromEmail">
  <wsdl:soap:operation soapAction="" />
  <wsdl:input name="getCourseFromEmailRequest">
    <wsdl:soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="getCourseFromEmailResponse">
    <wsdl:soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="cancel">
  <wsdl:soap:operation soapAction="" />
  <wsdl:input name="cancelRequest">
    <wsdl:soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="cancelResponse">
    <wsdl:soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>

```

```

<wsdl:service name="ApplyPhysicService">
  <wsdl:port binding="intf:ApplyPhysicSoapBinding" name="ApplyPhysic">
    <wsdl:soap:address
      location="http://161.246.6.91:9080/ApplyPhysic/services/ApplyPhysic"/>
    </wsdl:port>
  </wsdl:service>

```

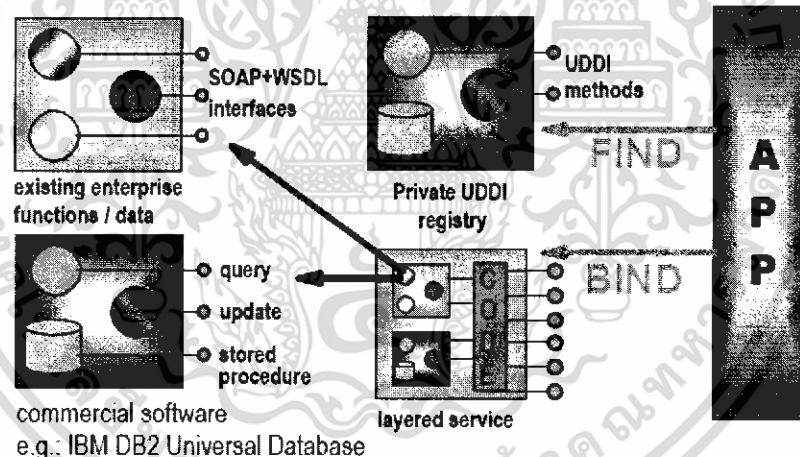
```
</wsdl:definitions>
```

2.6. UDDI (Universal Description, Discovery and Integration protocol)

UDDI (Universal Description, Discovery, and Integration) เป็นมาตรฐานที่ให้ชุดพื้นฐาน APIs (Application Programming Interface) ของ SOAP ที่สามารถนำมาใช้ในการพัฒนาเป็นตัวแทนของผู้ให้บริการ (Service broker) UDDI ใช้สำหรับค้นหา Service ที่ต้องการและเมื่อได้มาแล้ว UDDI ยังจัดหาข้อตกลงในวิธีการที่จะใช้งานเปรียบได้กับสมุดหน้าเหลือง เป็นมาตรฐานที่จัดตั้งขึ้นโดยบริษัทไอบีเอ็ม บริษัทไมโครซอฟต์ และบริษัทออริบา (Ariba) ปัจจุบันมีบริษัทที่ร่วมกันกำหนดมาตรฐานของ UDDI มากกว่า 70 บริษัท ซึ่งมาตรฐานของ UDDI ถูกกำหนดให้เป็นมาตรฐานสำหรับ B2B interoperability

UDDI for EAI Use

- SOAP+WSDL is a useful strategy for application or data integration with loose coupling
 - Especially helpful in mergers and acquisitions
 - Often fits into existing architectures, making them more flexible and adaptable
- Private UDDI can organize services in your enterprise



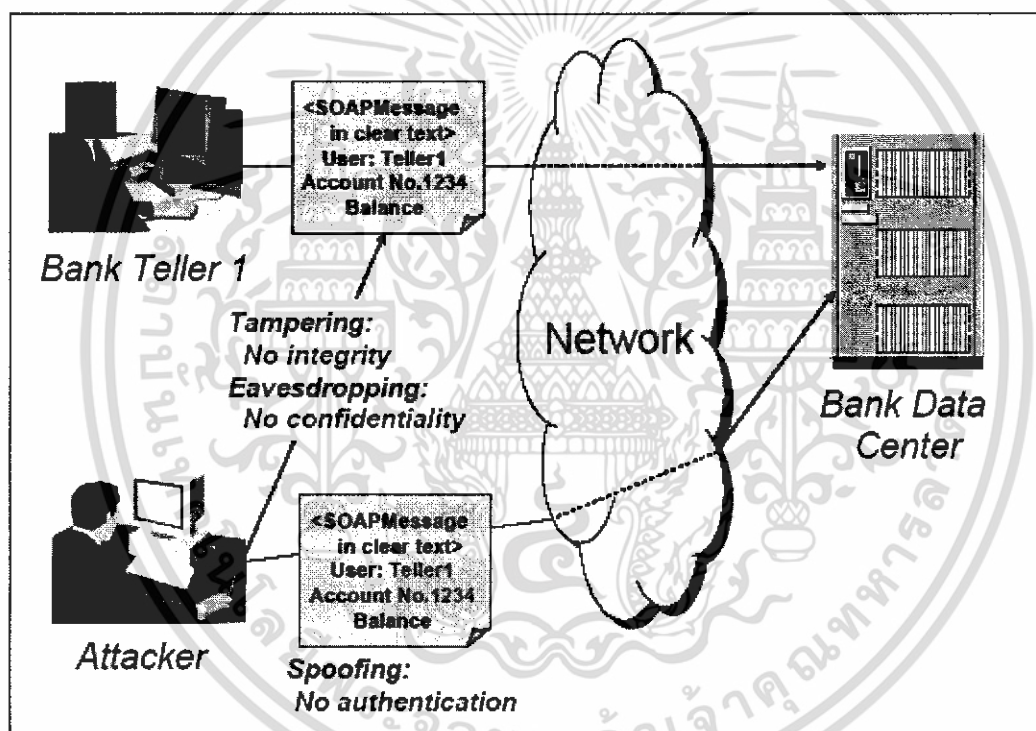
รูปที่ 2.12 แสดงการทำงานของ UDDI

บทที่ 3

Web Service Security (WS-Security)

3.1. Web Service Security (WS-Security)

WS-Security ออกแบบมาเพื่อเพิ่มความปลอดภัยของข้อมูลในระดับ Message ในการส่งข้อมูลระหว่างเว็บเซอร์วิส แบบ End-to-End และเพิ่มประสิทธิภาพในการส่ง SOAP Message ให้มีความน่าเชื่อถือ มีความถูกต้อง และ มีความปลอดภัย (integrity, confidentiality, authentication) ทำให้ข้อมูลมีความปลอดภัยตลอดเวลาไม่ว่าจะถูกส่งไปให้ Service ไหนก็ตาม



รูปที่ 3.1 แสดงตัวอย่างการส่งข้อมูลผ่านทาง เว็บเซอร์วิส ที่ไม่มีกาจัดการเรื่อง Security จากรูปสังเกตได้ว่าสามารถเกิดความเสี่ยงในการที่จะถูกขโมยหรือแก้ไขข้อมูลได้ 3 อย่าง

1. Spoofing: No Authentication

บุคคลที่ 3 สามารถดักจับ SOAP และ แก้ไขข้อมูล SOAP แล้วส่งให้ Service Provider พร้อมทั้ง สามารถปลอมเป็นเจ้าหน้าที่ของธนาคาร ได้

**** แก้ไขได้โดยการใช้การพิสูจน์ตัวตน (Authentication)**

2. Tempering: No Integrity

บุคคลที่ 3 สามารถดัก SOAP Packet ระหว่าง เว็บเซอร์วิส Client กับ เว็บเซอร์วิส Server ได้ สามารถแก้ไขข้อมูล บัญชีธนาคาร, รหัส ATM, ตัวเลขเงินในบัญชี เว็บเซอร์วิส Server ไม่มีการตรวจสอบว่า ข้อมูลที่ได้รับมานั้น ถูกต้องจริงหรือไม่

**** แก้ไขได้โดยการใช้การรักษาความสมบูรณ์ของข้อมูล (Integrity)**

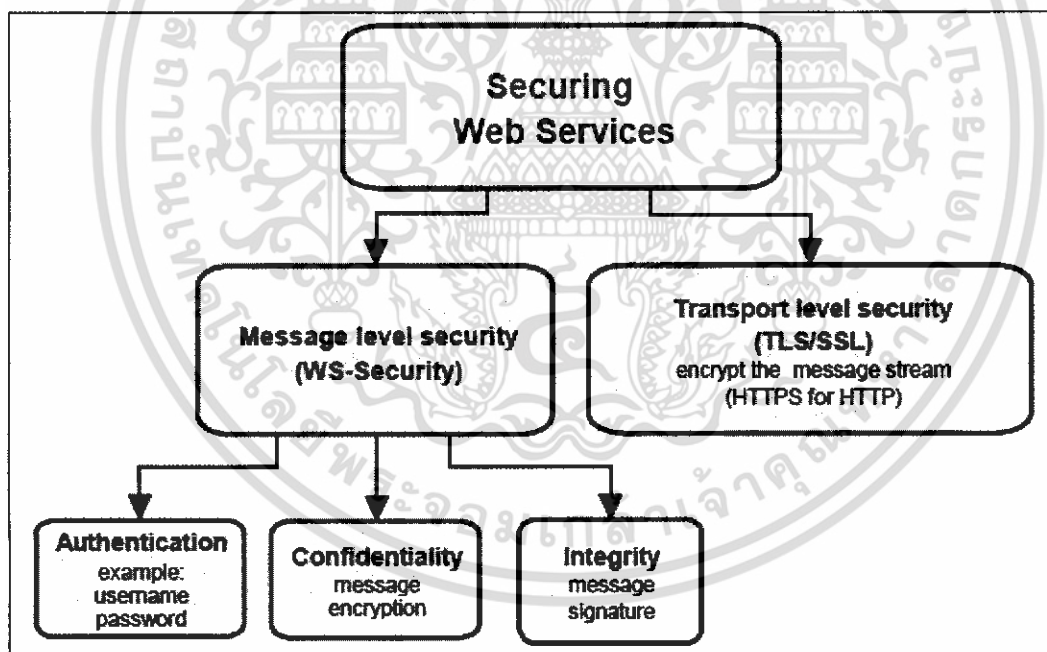
3. Eavesdropping: No Confidentiality

บุคคลที่ 3 สามารถดัก SOAP Packet และ ขโมยข้อมูลไปได้ทั้งหมด เนื่องจากข้อมูลไม่มีการเข้ารหัส ทำให้ข้อมูลที่เป็นความลับของลูกค้าถูกขโมยได้

**** แก้ไขได้โดยการใช้การเข้ารหัสข้อมูลเข้ามาช่วย (Encryption)**

จากตัวอย่างเหตุการณ์ที่เกิดขึ้นทำให้มีการป้องกัน Web Service Environment ดังนี้

- Message Level Security : Web Service Security (WS-Security)
- Transport Level Security : TLS/SSL



รูปที่ 3.2 แสดงโมเดล Securing Web Services

3.2. การเพิ่มความปลอดภัยในการพิสูจน์ตัวตน (Authentication)

การเพิ่มความปลอดภัยในการพิสูจน์ตัวตนคือขั้นตอนการยืนยันความถูกต้องของหลักฐาน (Identity) ที่แสดงว่าเป็นบุคคลที่กล่าวอ้างจริง การพิสูจน์ตัวตนมีหลายแบบ ไม่ว่าจะเป็น Username

Token , digital signature ,ID Assertion , LTPA หรือ Token อื่นๆ ในที่นี้จะกล่าวถึงUsername Token โดยเป็นการส่ง username password ไปกับ SOAP ด้วยเพื่อพิสูจน์ตัวตนของผู้ส่ง เมื่อ เว็บเซอร์วิส Server ได้รับ username และ password แล้วจะทำการตรวจสอบว่าตรงกับข้อมูลที่มีอยู่หรือไม่ ถ้าสิ่งที่ได้รับมาถูกต้อง Server จะรับ Message และ ทำงานต่อไป

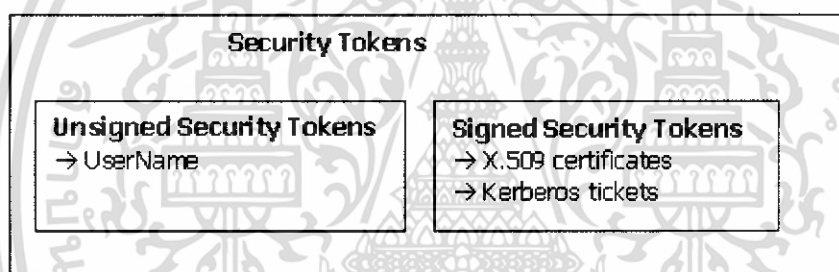
Token แบ่งเป็น 2 รูปแบบ คือ

Unsigned Security Token

เป็น Token ที่ไม่มีผู้รับรอง เป็นการแสดงสิทธิของตนเองให้เห็นเท่านั้น ไม่มีการรับรองแต่อย่างใด เช่น Username Token เป็นต้น

Signed Security Token

เป็น Token ที่มีการรับรองโดยผู้ที่มีความน่าเชื่อถือ เช่น X509 Certificate หรือ Kerberos Ticket เป็นต้น

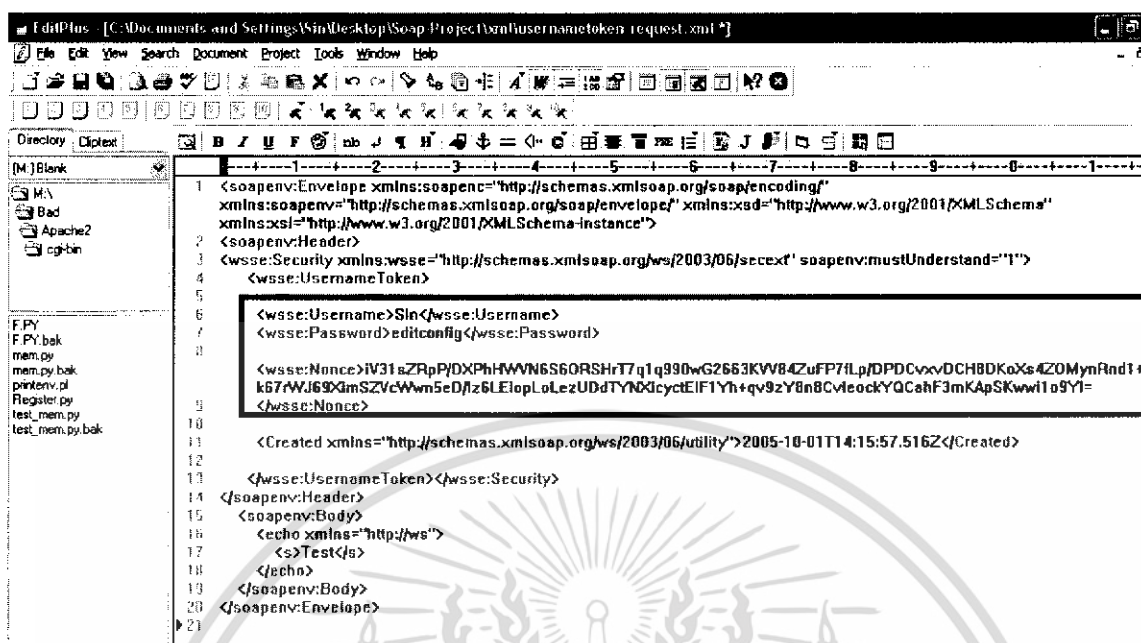


รูปที่ 3.3 แสดงประเภทของ Token

หลักการทำงานของ Username Token

Username Token แบบใช้ Nonce เป็นวิธีที่มีประสิทธิภาพมากวิธีหนึ่งในการตอบโต้การโจมตี Message โดยจะสร้าง cryptographic token แบบสุ่มเพื่อหลีกเลี่ยงการขโมย username token ใน SOAP Message โดยใช้คู่กับ ฟังก์ชัน basicauth แต่ Server จะต้องมีการมี cache ที่มีขนาดใหญ่ไว้ด้วย โดยจะนำมาใช้งานร่วมกับ timestamp ซึ่งจะช่วยให้ Server สามารถจำกัด cache ของ Nonce ให้อยู่ในช่วง “freshness”

เมื่อมีการส่ง Username token จากระบบหนึ่งไปอีกระบบหนึ่งโดยไม่มีการรักษาความปลอดภัยในการขนส่งข้อมูลเช่น HTTP อาจทำให้ token ถูกดักจับและถูกโจมตีกลับได้ key ที่ส่งไปอาจถูกนำกลับมาใช้ใหม่ทำให้เป็นจุดอ่อนในการโจมตีระบบได้ เว้นแต่จะใช้ XML digital signature หรือ XML Encryption เข้ามาช่วย



รูปที่ 3.4 แสดงตัวอย่าง SOAP ที่มีการทำ Username Token แล้ว

3.3. การรักษาความสมบูรณ์ของข้อมูล (Integrity)

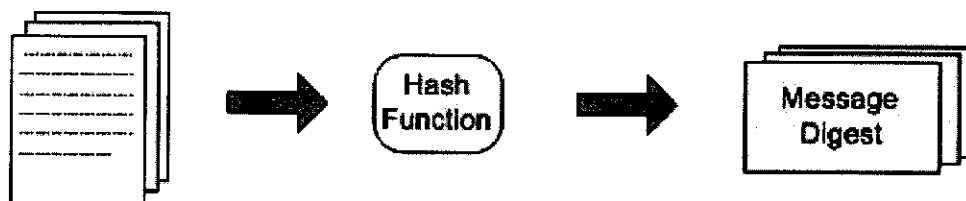
การรักษาความสมบูรณ์ คือการรับรองว่าข้อมูลจะไม่ถูกเปลี่ยนแปลงหรือทำลายไปจากต้นฉบับ ไม่ว่าจะเป็นโดยบังเอิญหรือดัดแปลงโดยเจตนาที่อาจส่งผลกระทบต่อข้อมูล การดูความสมบูรณ์ของข้อมูลคือการที่บุคคลที่ไม่ได้รับอนุญาตสามารถที่จะเข้าควบคุมการจัดการของข้อมูลได้ ในที่นี้จะกล่าวถึงการทำ XML digital signature บน SOAP Message ซึ่งเป็นการทำลายเซ็นไว้บน SOAP Message ซึ่งถ้า SOAP Message ถูกแก้ไขระหว่างที่ส่งข้อมูล ข้อมูลที่ถูกส่งไปจะทำให้กับผู้รับรู้ทันทีว่าถูกแก้ไขมาแล้ว

หลักการทางาน และตัวอย่าง XML Digital Signature

XML Digital Signature เป็นการระบุลายเซ็นดิจิทัลลงบน XML โดยใช้หลักการ Cryptographic ที่มีอยู่แล้ว เช่น RSA, HMAC, SHA1 ช่วยในการทำให้ผู้รับ Message สามารถมั่นใจได้ว่าไม่มีการขโมยข้อมูลหรือข้อผิดพลาดใดๆเกิดขึ้นใน Message หลังจากที่ได้มีการ sign key ไปแล้ว หรือเรียกได้ว่าเป็น proof of possession

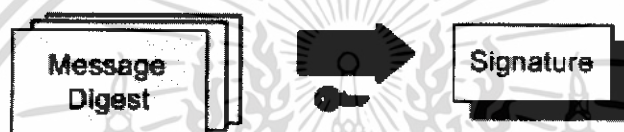
Digital Signature เป็นการนำหลักการของการทำงานของการเข้ารหัสแบบใช้คู่รหัสกุญแจเพื่อการพิสูจน์ตัวตนมาประยุกต์ใช้ ระบบของลายเซ็นดิจิทัลสามารถแบ่งเป็นขั้นตอนได้ดังนี้

1. เมื่อผู้ส่งต้องการจะส่งข้อมูลไปยังผู้รับ ข้อมูลนั้นจะถูกนำไปเข้าฟังก์ชันทางคณิตศาสตร์ที่เรียกว่า "แฮชฟังก์ชัน" ได้เมสเชสไดเจสต์ (Message Digest) ออกมา



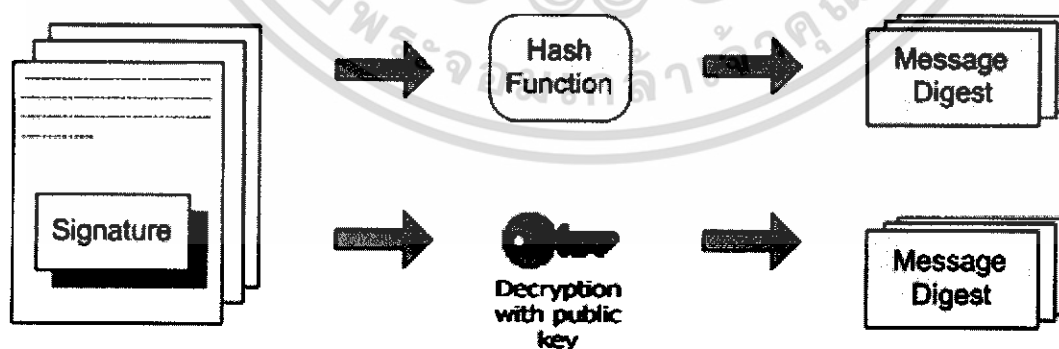
รูปที่ 3.5 การส่งข้อมูลเข้าไปใน Hash function

2. การใช้กุญแจส่วนตัวเข้ารหัสข้อมูล หมายถึงว่าผู้ส่งได้ลงลายเซ็นดิจิทัล ยินยอมที่จะให้ผู้รับสามารถทำการตรวจสอบด้วยกุญแจสาธารณะของผู้ส่งเพื่อพิสูจน์ตัวตนของผู้ส่งได้



รูปที่ 3.6 การเข้ารหัสเมสเสจไดเจสต์ด้วยกุญแจส่วนตัวเพื่อเป็นการลงลายเซ็น

3. การตรวจสอบข้อมูลว่าถูกส่งมาจากผู้ส่งคนนั้นจริงในด้านผู้รับ โดยการนำข้อมูลมาผ่านแฮชฟังก์ชันเพื่อคำนวณค่าเมสเสจไดเจสต์ และถอดรหัสลายเซ็นอิเล็กทรอนิกส์ด้วยกุญแจสาธารณะของผู้ส่ง ถ้าสามารถถอดได้อย่างถูกต้อง จะเป็นการยืนยันข้อมูลจากผู้ส่งคนนั้นจริง และถ้าข้อมูลเมสเสจไดเจสต์ที่ได้จากการถอดรหัสเท่ากับค่าเมสเสจไดเจสต์ในตอนต้นที่ทำการคำนวณได้ จะถือว่าข้อมูลดังกล่าวนั้นถูกต้อง



รูปที่ 3.7 ขั้นตอนการเปรียบเทียบความถูกต้อง

ลายเซ็นอิเล็กทรอนิกส์นิยมนำไปใช้ในระบบรักษาความปลอดภัยในการชำระเงินผ่านระบบ
อินเทอร์เน็ต ซึ่งในปัจจุบันนี้การทำธุรกรรมการเงินอิเล็กทรอนิกส์ได้รับความนิยมเป็นอย่างมาก

10/3/2005 11:25 AM

```

1 <soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
2 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
3 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4 xmlns:xs="http://www.w3.org/2001/XMLSchema-instance">
5   <soapenv:Header>
6     <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext"
7       soapenv:actor="myActorURL" soapenv:mustUnderstand="1">
8       <wsse:BinarySecurityToken xmlns:wssu="http://schemas.xmlsoap.org/ws/2003/06/utility"
9         EncodingType="wsse:Base64Binary" ValueType="wsse:X509v3"
10        wsu:id="wsssecurity_binary_security_token_id_408039426707555435">
11          MIIIDQTCACgppqVIBAgICAQQwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBmMCSAAETAP
12          BgnNVBAgTECEhbmFnZXRlbnMQwwCgYDVQQKEwQxODDAKBoNVAATAIJRSTDEQMAGAUUE
13          AUMHSW50IENBMJAeFwOWMTcWEwMDcwTUU0MDZAFwOUMTEwMDcwTUU0MDZAMFQxCzAIBGVN
14          BAYTAkppQURRZWdhdyDQYQJEWLW5WShZF3YTEMMABGUAIECHMODSUNNMQwwCgYDVQQLEwNU
15          UkwkfjAUBGNVBAITdVNFQPBSZCFIZONODZIKwQZhwDQYJKoZIhvcNAQEBBQADgQYDARIEG)
16       </wsse:BinarySecurityToken>
17       <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
18         <SignedInfo>
19           <CanonicalizationMethod
20             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></CanonicalizationMethod>
21           <SignatureMethod
22             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></SignatureMethod>
23           <Reference URI="#wsssecurity_body_id_4508207003845519626">
24             <Transforms>
25               <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
26                 <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
27                   PrefixList="xs soapenc xsd # default wssu soapenv"></ec:InclusiveNamespaces>
28               </Transform>
29             </Transforms>
30             <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></DigestMethod>
31             <DigestValue>zp0qJanV9BKA61SI6AZA0qpPQ= </DigestValue>
32           </Reference>
33         </SignedInfo>
34         <SignatureValue>
35           g1Ez2tLQMPDCL4OLktbLLpfAw77XJez9GoJaLnT7zenPRjrHmMgmNLfNS
36           v/VW0SqTYfhX6vJMukPFui83JTwdPeTs/BofWkr/War6VdkSS4UicPd9
37           VcORCC+zkgVzfzc533+EbVixAKVhSS6147pGUGENauEMXXZWT8=
38         </SignatureValue>
39       </Signature>
40     </wsse:Security>
41   </soapenv:Header>
42   <soapenv:Body xmlns:wssu="http://schemas.xmlsoap.org/ws/2003/06/utility"
43     wsu:id="wsssecurity_body_id_4508207003845519626"><echo xmlns="http://ws"><test
44     Message</s></echo>
45   </soapenv:Body>
46 </soapenv:Envelope>

```

รูปที่ 3.8 แสดงตัวอย่าง SOAP Message ที่มีการใช้ Digital signature

3.4. การเพิ่มความปลอดภัยของข้อมูลที่เป็นความลับ (Confidentiality)

การเพิ่มความปลอดภัยของข้อมูลที่เป็นความลับ คือการรับรองว่าจะมีการเก็บข้อมูลไว้เป็นความลับ และ ผู้มีสิทธิเท่านั้นจึงจะเข้าถึงข้อมูลนั้นได้ ในที่นี้จะกล่าวถึงการทำ XML Encryption บน SOAP Message หรือการเข้ารหัสข้อมูล

หลักการทางาน ของ XML Encryption

การทำ XML Encryption คล้ายกับการเข้ารหัสแบบต่างๆ ไป แต่แตกต่างที่ การเข้ารหัส XML สามารถเลือกทำการ เข้ารหัสได้ เฉพาะส่วน หรือเฉพาะ element ได้

3.4.1. อัลกอริทึมในการเข้ารหัสข้อมูลมี 2 ประเภทหลัก คือ

3.4.1.1. อัลกอริทึมแบบสมมาตร (Symmetric key algorithms)

อัลกอริทึมแบบนี้จะใช้กุญแจที่เรียกว่า กุญแจลับ (Secret key) ซึ่งมีเพียงหนึ่งเดียวเพื่อใช้ในการเข้ารหัสและถอดรหัสข้อความที่ส่งไป อัลกอริทึมยังสามารถแบ่งย่อยออกเป็น 2 ประเภท ได้แก่ แบบบล็อก (Block Algorithms) ซึ่งจะทำให้การเข้ารหัสทีละบล็อก (1 บล็อกประกอบด้วยหลายไบต์ เช่น 64 ไบต์ เป็นต้น) และแบบสตรีม (Stream Algorithms) ซึ่งจะทำให้การเข้ารหัสทีละไบต์ อัลกอริทึมแบบนี้จะใช้กุญแจที่เรียกว่า กุญแจลับ (Secret key) ซึ่งมีเพียงหนึ่งเดียวเพื่อใช้ในการเข้ารหัสและถอดรหัสข้อความที่ส่งไป อัลกอริทึมยังสามารถแบ่งย่อยออกเป็น 2 ประเภท ได้แก่ แบบบล็อก (Block Algorithms) ซึ่งจะทำให้การเข้ารหัสทีละบล็อก (1 บล็อกประกอบด้วยหลายไบต์ เช่น 64 ไบต์ เป็นต้น) และแบบสตรีม (Stream Algorithms) ซึ่งจะทำให้การเข้ารหัสทีละไบต์

3.4.1.2. อัลกอริทึมแบบอสมมาตร (Asymmetric key algorithms)

อัลกอริทึมนี้จะใช้กุญแจสองตัวเพื่อทำงาน ตัวหนึ่งใช้ในการเข้ารหัสและอีกตัวหนึ่งใช้ในการถอดรหัสข้อมูลที่เข้ารหัสมาโดยกุญแจตัวแรก อัลกอริทึมกลุ่มสำคัญในแบบอสมมาตรนี้คือ อัลกอริทึมแบบกุญแจสาธารณะ (Public keys Algorithms) ซึ่งใช้กุญแจที่เรียกกันว่า กุญแจสาธารณะ (Public keys) ในการเข้ารหัสและใช้กุญแจที่เรียกกันว่า กุญแจส่วนตัว (Private keys) ในการถอดรหัสข้อมูลนั้น กุญแจสาธารณะนี้สามารถส่งมอบให้กับผู้อื่นได้ เช่น เพื่อนร่วมงานที่เราต้องการติดต่อด้วย หรือแม้กระทั่งวางไว้บนเว็บไซต์เพื่อให้ผู้อื่นสามารถดาวน์โหลดไปใช้งานได้ สำหรับกุญแจส่วนตัวนั้นต้องเก็บไว้กับผู้ใช้เป็นเจ้าของกุญแจส่วนตัวเท่านั้นและห้ามเปิดเผยให้ผู้อื่นทราบโดยเด็ดขาด

3.4.2. ปัญหาของอัลกอริทึมแบบสมมาตร (Symmetric key Problem)

อัลกอริทึมแบบสมมาตรมีความสำคัญไม่ด้อยไปกว่าอัลกอริทึมแบบอสมมาตร ทั้งนี้เนื่องจากอัลกอริทึมแบบแรกทำงานได้รวดเร็วและง่ายต่อการใช้งานกว่าแบบหลัง อย่างไรก็ตามอัลกอริทึมแบบสมมาตรยังมีปัญหาที่สำคัญ 3 ประการ ซึ่งเป็นข้อจำกัดในการใช้งานอัลกอริทึมนี้

1. ในการใช้งานอัลกอริทึมนี้ สองกลุ่มที่ต้องการแลกเปลี่ยนข้อมูลกัน (เช่น องค์กร ก และ ข) จำเป็นต้องแลกเปลี่ยนกุญแจลับกันก่อน (ซึ่งอาจหมายถึงส่งมอบกุญแจลับให้กับอีกกลุ่มหนึ่ง) การแลกเปลี่ยนกุญแจนั้นอาจทำได้อย่างยุ่งยากและไม่สะดวก

2. ทั้งสองกลุ่มต้องรักษากุญแจลับนั้นไว้เป็นอย่างดี ห้ามเปิดเผยให้ผู้อื่นล่วงรู้โดยเด็ดขาด การที่กุญแจถูกเปิดเผยออกไปสู่ผู้อื่น (จะโดยกลุ่มใดกลุ่มหนึ่งก็ตาม) และอีกกลุ่มหนึ่งไม่ได้รับทราบปัญหานี้ อาจก่อให้เกิดปัญหาให้กับกลุ่มที่ไม่ทราบนี้ได้ เช่น กลุ่มนี้อาจส่งข้อความที่เป็นความลับไปให้กับอีกกลุ่มหนึ่ง แต่ข้อความนี้อาจถูกเปิดเผยได้โดยใช้กุญแจลับที่ล่วงรู้โดยผู้อื่น

3. สำหรับสองกลุ่มที่ต้องการติดต่อกัน จำเป็นต้องใช้กุญแจลับเป็นจำนวน 1 กุญแจเพื่อติดต่อกัน สมมติว่ามีผู้ที่ต้องติดต่อกันเป็นจำนวน n กลุ่ม จำนวนกุญแจลับทั้งหมดที่ต้องแลกเปลี่ยนกันคิดเป็นจำนวนทั้งหมด C_{2n} หรือเท่ากับ $n(n-1)/2$ กุญแจ ซึ่งจะเห็นได้ว่าจำนวนกุญแจมีมากมายเกินไป ซึ่งอาจก่อให้เกิดปัญหาด้านการรักษาความปลอดภัยให้กับกุญแจเหล่านี้

อัลกอริทึมแบบกุญแจสาธารณะ (ซึ่งเป็นแบบอสมมาตร) ช่วยแก้ปัญหาเหล่านี้ได้ทั้งหมด ผู้ใช้ที่ถือกุญแจส่วนตัวและต้องการให้บุคคลอื่นที่ตนติดต่อด้วยส่งเอกสารหรือข้อความที่เข้ารหัสมาหาตนสามารถเผยแพร่กุญแจสาธารณะของตนไว้บนเว็บไซต์หรือในที่สาธารณะซึ่งผู้อื่นสามารถเข้ามาดาวน์โหลดไปใช้งานได้ วิธีการใช้งานคือให้บุคคลอื่นที่มาดาวน์โหลดกุญแจไปนั้นทำการเข้ารหัสข้อความที่ต้องการส่งด้วยกุญแจสาธารณะ แล้วจึงส่งข้อความที่เข้ารหัสไปให้กับผู้เป็นเจ้าของกุญแจสาธารณะ โดยวิธีนี้จะมีผู้อื่นสามารถเปิดดูข้อความที่เข้ารหัสนั้นได้ยกเว้นผู้ที่ถือกุญแจส่วนตัว (ที่เป็นคู่ของกุญแจสาธารณะนั้น) จึงจะสามารถเปิดข้อความนี้ได้

การเผยแพร่กุญแจสาธารณะในสถานที่ต่างๆ ได้ช่วยลดความยุ่งยากในการแลกเปลี่ยนกุญแจกันซึ่งเป็นปัญหาข้อแรกของการเข้ารหัสแบบสมมาตร สำหรับปัญหาที่ว่าทั้งสองกลุ่มจะต้องรักษากุญแจลับไว้เป็นอย่างดีนั้น วิธีการของกุญแจสาธารณะจะทำให้ผู้ที่ต้องรับผิดชอบเหลือเพียงผู้เดียว กล่าวคือ ผู้ถือกุญแจส่วนตัว ซึ่งห้ามให้ผู้อื่นล่วงรู้โดยเด็ดขาด

สำหรับปัญหาที่สามที่ว่าจำนวนกุญแจลับที่จำเป็นต้องใช้มีมากมายเกินไป วิธีการของกุญแจสาธารณะจะใช้จำนวนกุญแจที่ประหยัดกว่า เนื่องจากกุญแจสาธารณะ 1 กุญแจของกลุ่มๆ หนึ่งจะสามารถเผยแพร่ให้กับกี่กลุ่มก็ได้ที่เราต้องการติดต่อด้วย (แทนที่จะเป็น 1 กุญแจลับต่อสองกลุ่มที่ต้องการติดต่อกัน) ดังนั้นถ้ามีกลุ่มที่ต้องติดต่อกันจำนวน n กลุ่ม จำนวนกุญแจส่วนตัวที่ต้องระวังกวาก็คือ n กุญแจ ซึ่งจะเห็นได้ว่าลดลงไปได้เป็นจำนวนมาก

ข้อเสียที่สำคัญของระบบกุญแจสาธารณะที่สำคัญคือ ต้องใช้เวลาในการคำนวณการเข้ารหัสและถอดรหัส เมื่อเทียบกับระบบกุญแจสมมาตร และอาจใช้เวลาเป็นพันเท่าของเวลาที่ใช้โดยระบบกุญแจสมมาตร

3.4.3. ความแข็งแกร่งของอัลกอริทึมสำหรับการเข้ารหัส

ความแข็งแกร่งของอัลกอริทึมหมายถึงความยากในการที่ผู้บุกรุกจะสามารถถอดรหัสข้อมูลได้โดยปราศจากกุญแจที่ใช้ในการเข้ารหัส ซึ่งจะขึ้นอยู่กับปัจจัยดังนี้

1. การเก็บกุญแจเข้ารหัสไว้อย่างเป็นความลับ ผู้เป็นเจ้าของกุญแจลับหรือส่วนตัวต้องระมัดระวังไม่ให้กุญแจสูญหายหรือล่วงรู้โดยผู้อื่น

2. ความยาวของกุญแจเข้ารหัส ปกติกุญแจเข้ารหัสจะมีความยาวเป็นบิต ยิ่งจำนวนบิตของกุญแจยิ่งมาก ยิ่งทำให้การเดาเพื่อค้นหากุญแจที่ถูกต้องเป็นไปได้ยากยิ่งขึ้น (เช่น กุญแจขนาด 1 บิต จะสามารถแทนตัวเลขได้ 2 ค่าคือ 0 กับ 1 กุญแจขนาด 2 บิต จะเป็นไปได้ 4 ค่าคือ 0, 1, 2, 3 เป็นต้น)

3. ความไม่เกรงกลัวต่อการศึกษาหรือดูอัลกอริทึมเพื่อหารูปแบบของการเข้ารหัส อัลกอริทึมที่ดีต้องเปิดให้ผู้รู้ทำการศึกษาในรายละเอียดได้โดยไม่เกรงว่าผู้ศึกษาจะสามารถจับรูปแบบของการเข้ารหัสได้

4. การมีประตูลับในอัลกอริทึม อัลกอริทึมที่ดีต้องไม่แฝงไว้ด้วยประตูลับที่สามารถใช้ป้อนทางเข้าไปสู่อัลกอริทึม แล้วอาจใช้เพื่อทำการถอดรหัสข้อมูลได้ ประตูลับนี้ทำให้ไม่จำเป็นต้องใช้กุญแจในการถอดรหัส

5. ความไม่เกรงกลัวต่อปัญหาการหาความสัมพันธ์ในข้อมูลที่ได้รับ กล่าวคือเมื่อผู้บุกรุกทราบข้อมูลบางอย่างที่เป็นข้อมูลตั้งต้นซึ่งยังไม่ได้เข้ารหัส รวมทั้งมีข้อมูลที่เข้ารหัสแล้ว (ของข้อมูลตั้งต้นนั้น) ผู้บุกรุกอาจจะสามารถหาความสัมพันธ์ระหว่างข้อความทั้งสองนั้นได้ ซึ่งจะเป็นวิธีการในการถอดรหัสข้อมูลได้ ปัญหานี้เรียกกันว่า Known plaintext attack (คำว่า plaintext หมายถึงข้อความตั้งต้นที่ยังไม่ได้ผ่านการเข้ารหัส)

คุณสมบัติของข้อความตั้งต้น คุณสมบัตินี้อาจใช้เป็นช่องทางในการถอดรหัสข้อมูลได้ อัลกอริทึมที่ดีต้องไม่ใช้คุณสมบัติของข้อความปกในการเข้ารหัสข้อมูล

คำแนะนำในการเลือกใช้อัลกอริทึมคือให้ใช้อัลกอริทึมที่ได้มีการใช้งานมาเป็นระยะเวลานานแล้ว ทั้งนี้เนื่องจากหากปัญหาของอัลกอริทึมนี้มีจริง ก็คงเกิดขึ้นมานานแล้วและก็คงเป็นที่ทราบกันแล้ว นั่นคืออย่างน้อยที่สุดจวบจนกระทั่งถึงปัจจุบัน ก็ยังไม่มีผู้บุกรุกที่ทำให้อัลกอริทึมนั้นไม่สามารถใช้งานได้อย่างปลอดภัยเป็นที่ประจักษ์ ดังนั้นจึงไม่ควรใช้อัลกอริทึมใหม่ๆ ที่เพิ่งได้มีการนำเสนอเข้าสู่สาธารณะ เพราะอาจมีช่องโหว่แฝงอยู่และยังไม่มีที่ทราบในขณะนี้

3.4.4. ความยาวของกุญแจที่ใช้ในการเข้ารหัส

ความยาวของกุญแจเข้ารหัสมีหน่วยนับเป็นบิต หนึ่งบิตในคอมพิวเตอร์เป็นตัวเลขฐานสองที่ประกอบด้วยค่า 0 และ 1 กุญแจที่มีความยาว 1 บิต ตัวเลขที่เป็นไปได้เพื่อแทนกุญแจนั้น จึงอาจมีค่าเป็น 0 หรือ 1 กุญแจที่มีความยาว 2 บิต ตัวเลขที่เป็นไปได้จึงเป็น 0, 1, 2 และ 3 ตามลำดับ กุญแจที่มีความยาว 3 บิต ตัวเลขที่เป็นไปได้จะอยู่ระหว่าง 0 ถึง 7 ดังนั้นเมื่อเพิ่มความยาวของกุญแจทุกๆ 1 บิต ค่าที่เป็นไปได้ของกุญแจจะเพิ่มขึ้นเป็นสองเท่าตัว หรือจำนวนกุญแจที่เป็นไปได้จะเพิ่มขึ้นเป็น 2 เท่าตัวนั่นเอง

ฉะนั้นจะเห็นได้ว่ากุญแจยิ่งมีความยาวมาก โอกาสที่ผู้บุกรุกจะสามารถคาดเดากุญแจที่ตรงกับหมายเลขที่ถูกต้องของกุญแจจะยิ่งยากมากขึ้นตามลำดับ ในการที่ผู้บุกรุกลองผิดลองถูกกับกุญแจโดยใช้กุญแจที่มีหมายเลขต่างๆ กัน เพื่อหวังที่จะพบกุญแจที่ถูกต้องและสามารถใช้ออกรหัสข้อมูลได้ การลองผิดลองถูกนี้เราเรียกกันว่า Key search หรือการค้นหากุญแจนั่นเอง ทฤษฎีได้กล่าวไว้ว่าการลองผิดลองถูกนี้โดยเฉลี่ยจะต้องทดลองกับกุญแจเป็นจำนวนครึ่งหนึ่งของกุญแจทั้งหมดก่อนที่จะพบกุญแจที่ถูกต้อง

ความยาวของกุญแจที่มีขนาดเหมาะสมจึงขึ้นอยู่กับความเร็วในการค้นหากุญแจของผู้บุกรุก และระยะเวลาที่ต้องการให้ข้อมูลมีความปลอดภัย ตัวอย่างเช่น ถ้าผู้บุกรุกสามารถลองผิดลองถูกกับกุญแจเป็นจำนวน 10 กุญแจภายในหนึ่งวินาทีแล้ว กุญแจที่มีความยาว 40 บิต จะสามารถป้องกันข้อมูลไว้ได้ 3,484 ปี ถ้าผู้บุกรุกสามารถลองได้เป็นจำนวน 1 ล้านกุญแจในหนึ่งวินาที (เทคโนโลยีปัจจุบันสามารถทำได้) กุญแจที่มีความยาว 40 บิตจะสามารถป้องกันข้อมูลไว้ได้เพียง 13 วันเท่านั้น (ซึ่งอาจไม่เพียงพอสำหรับในบางลักษณะงาน) ด้วยเทคโนโลยีในปัจจุบันหากผู้บุกรุกสามารถทดลองได้เป็นจำนวน 1,000 ล้านกุญแจในหนึ่งวินาที กุญแจขนาด 128 บิตจะสามารถป้องกันข้อมูลไว้ได้ 1022 ปี ดังนั้นด้วยลักษณะงานทั่วไปกุญแจขนาด 128 บิตจะพอเพียงต่อการรักษาความลับของข้อมูลเอาไว้ได้

3.4.5. อัลกอริทึมสำหรับการเข้ารหัสแบบสมมาตร

อัลกอริทึมสำหรับการเข้ารหัสแบบสมมาตรในปัจจุบันมีเป็นจำนวนมาก ข้างล่างนี้จะนำเสนอเพียงจำนวนหนึ่งซึ่งเป็นอัลกอริทึมที่เป็นที่รู้จักกันดีในวงการของการเข้ารหัสข้อมูล

อัลกอริทึม DES

DES ย่อมาจาก Data Encryption Standard เป็นอัลกอริทึมแบบบล็อกซึ่งใช้กุญแจที่มีขนาดความยาว 56 บิตและเป็นอัลกอริทึมที่มีความแข็งแกร่ง แต่เนื่องด้วยขนาดความยาวของกุญแจที่มีขนาดเพียง 56 บิต ซึ่งในปัจจุบันถือว่าสั้นเกินไป ผู้บุกรุกอาจใช้วิธีการลองผิดลองถูกเพื่อค้นหากุญแจที่ถูกต้องสำหรับการถอดรหัสได้

อัลกอริทึม Triple-DES

Triple-DES เป็นอัลกอริทึมที่เสริมความปลอดภัยของ DES ให้มีความแข็งแกร่งมากขึ้นโดยใช้ อัลกอริทึม DES เป็นจำนวนสามครั้งเพื่อทำการเข้ารหัส แต่ครั้งจะใช้กุญแจในการเข้ารหัสที่แตกต่าง กัน ดังนั้นจึงเปรียบเสมือนการใช้กุญแจเข้ารหัสที่มีความยาวเท่ากับ $56 \times 3 = 168$ บิต Triple-DES ได้ถูก ใช้งานกับสถาบันทางการเงินอย่างแพร่หลาย รวมทั้งใช้งานกับโปรแกรม Secure Shell (ssh) ด้วย

อัลกอริทึม Rijndael (หรืออัลกอริทึม AES)

อัลกอริทึมนี้ได้รับการพัฒนาโดย Joan Daemen และ Vincent Rijmen ในปี 2000 อัลกอริทึม ได้รับการคัดเลือกโดยหน่วยงาน National Institute of Standard and Technology (NIST) ของ สหรัฐอเมริกาให้เป็นมาตรฐานในการเข้ารหัสชั้นสูงของประเทศ อัลกอริทึมมีความเร็วสูงและมีขนาด กะทัดรัดโดยสามารถใช้กุญแจที่มีความยาวขนาด 128, 192 และ 256 บิต

3.4.6. อัลกอริทึมสำหรับการเข้ารหัสแบบกุญแจสาธารณะ (หรือการเข้ารหัสแบบอสมมาตร)

อัลกอริทึม RSA

อัลกอริทึม RSA ได้รับการพัฒนาขึ้นที่มหาวิทยาลัย MIT ในปี 1977 โดยศาสตราจารย์ 3 คน ซึ่ง ประกอบด้วย Ronald Rivest, Adi Shamir และ Leonard Adleman ชื่อของอัลกอริทึมได้รับการตั้งชื่อ ตามตัวอักษรตัวแรกของนามสกุลของศาสตราจารย์ทั้งสามคน อัลกอริทึมนี้สามารถใช้ในการเข้ารหัส ข้อมูลรวมทั้งการลงลายมือชื่ออิเล็กทรอนิกส์ด้วย

อัลกอริทึม DSS

DSS ย่อมาจาก Digital Signature Standard อัลกอริทึมนี้ได้รับการพัฒนาขึ้นมาโดย National Security Agency ในประเทศสหรัฐอเมริกาและได้รับการรับรองโดย NIST ให้เป็นมาตรฐานกลาง สำหรับการลงลายมือชื่ออิเล็กทรอนิกส์ในประเทศสหรัฐอเมริกา

3.4.7. อัลกอริทึมสำหรับสร้างเมสเสจไดเจสต์

เมสเสจไดเจสต์ (Message Digest) หรือเรียกสั้นๆ ว่าไดเจสต์ แปลว่าข้อความสรุปจากเนื้อหา ข้อความตั้งต้น โดยปกติข้อความสรุปจะมีความยาวน้อยกว่าความยาวของข้อความตั้งต้นมาก จุดประสงค์สำคัญของอัลกอริทึมนี้คือ การสร้างข้อความสรุปที่สามารถใช้เป็นตัวแทนของข้อความตั้ง ต้นได้ โดยทั่วไปข้อความสรุปจะมีความยาวอยู่ระหว่าง 128 ถึง 256 บิต และจะไม่ขึ้นกับขนาดความ ยาวของข้อความตั้งต้น

3.4.8. คุณสมบัติที่สำคัญของอัลกอริทึมสำหรับสร้างไคเจสต์มีดังนี้

ทุกๆ บิตของไคเจสต์จะขึ้นอยู่กับทุกบิตของข้อความตั้งต้น ถ้าบิตใดบิตหนึ่งของข้อความตั้งต้นเกิดการเปลี่ยนแปลง เช่น ถูกแก้ไข ทุกๆ บิตของไคเจสต์จะมีโอกาสร้อยละ 50 ที่จะแปรเปลี่ยนค่าไปด้วย ซึ่งหมายถึงว่า 0 เปลี่ยนค่าเป็น 1 และ 1 เปลี่ยนเป็น 0

คุณสมบัติข้อนี้สามารถอธิบายได้ว่าการเปลี่ยนแปลงแก้ไขข้อความตั้งต้นโดยผู้ไม่ประสงค์ดี แม้ว่าอาจแก้ไขเพียงเล็กน้อยก็ตาม เช่น เพียง 1 บิตเท่านั้น ก็จะส่งผลให้ผู้รับข้อความทราบว่าข้อความที่ตนได้รับไม่ใช่ข้อความตั้งต้น (โดยการนำข้อความที่ตนได้รับเข้าอัลกอริทึมเพื่อทำการคำนวณหาไคเจสต์ออกมา แล้วจึงเปรียบเทียบไคเจสต์ที่คำนวณได้กับไคเจสต์ที่ส่งมาให้ด้วย ถ้าต่างกัน แสดงว่าข้อความที่ได้รับนั้นถูกเปลี่ยนแปลงแก้ไข)

โอกาสที่ข้อความตั้งต้น 2 ข้อความใดๆ ที่มีความแตกต่างกัน จะสามารถคำนวณได้ค่าไคเจสต์เดียวกันมีโอกาสน้อยมาก

คุณสมบัติข้อนี้ทำให้แน่ใจได้ว่า เมื่อผู้ไม่ประสงค์ดีทำการแก้ไขข้อความตั้งต้น ผู้รับข้อความที่ถูกแก้ไขไปแล้วนั้นจะสามารถตรวจพบได้ถึงความผิดปกติที่เกิดขึ้นอย่างแน่นอน

อย่างไรก็ตามในทางทฤษฎีแล้ว มีโอกาสที่ข้อความ 2 ข้อความที่แตกต่างกันจะสามารถคำนวณแล้วได้ค่าไคเจสต์เดียวกัน ปัญหานี้เรียกกันว่าการชนกันของไคเจสต์ (Collision) อัลกอริทึมสำหรับสร้างไคเจสต์ที่ดีควรมีโอกาสน้อยมากๆ ที่จะก่อให้เกิดปัญหาการชนกันของไคเจสต์ อัลกอริทึมสำหรับสร้างไคเจสต์ยอดนิยมมีดังนี้

อัลกอริทึม MD2

ผู้พัฒนาคือ Ronald Rivest อัลกอริทึมนี้เชื่อกันว่ามีความแข็งแกร่งที่สุดในบรรดาอัลกอริทึมต่างๆ ที่ Rivest พัฒนาขึ้นมา (ความแข็งแกร่งพิจารณาได้จากคุณสมบัติสามประการข้างต้น) ข้อเสียของอัลกอริทึมนี้คือใช้เวลามากในการคำนวณไคเจสต์หลายๆ MD2 จึงไม่ค่อยได้มีการใช้งานกันมากนัก MD2 สร้างไคเจสต์ที่มีความยาว 128 บิต

อัลกอริทึม MD4

ผู้พัฒนาคือ Rivest เช่นเดียวกับ MD2 อัลกอริทึมนี้พัฒนาขึ้นมาเพื่อแก้ปัญหาล่าช้าในการคำนวณของ MD2 อย่างไรก็ตามในภายหลังได้พบว่าอัลกอริทึมมีข้อบกพร่องที่เกี่ยวข้องกับคุณสมบัติข้อที่สามโดยตรง กล่าวคือปัญหาการชนกันของไคเจสต์มีโอกาสเกิดขึ้นได้ ซึ่งผู้บุกรุกอาจใช้ประโยชน์จากจุดอ่อนนี้เพื่อทำการแก้ไขข้อความตั้งต้นที่ส่งมาให้ได้ MD4 ผลิตไคเจสต์ที่มีขนาด 128 บิต

อัลกอริทึม MD5

Rivest เป็นผู้พัฒนาเช่นกันโดยพัฒนาต่อจาก MD4 เพื่อให้มีความปลอดภัยที่สูงขึ้น ถึงแม้จะเป็นที่นิยมใช้งานกันอย่างแพร่หลาย ทว่าในปี 1996 ก็มีผู้พบจุดบกพร่องของ MD5 (เช่นเดียวกับ MD4) จึงทำให้ความนิยมเริ่มลดลง MD5 ผลิตไคเจสต์ที่มีขนาด 128 บิต

อัลกอริทึม SHA

SHA ย่อจาก Secure Hash Algorithm อัลกอริทึม SHA ได้รับแนวคิดในการพัฒนามาจาก MD4 และได้รับการพัฒนาขึ้นมาเพื่อใช้งานร่วมกับอัลกอริทึม DSS (ซึ่งใช้ในการลงลายมือชื่ออิเล็กทรอนิกส์) หลังจากที่ได้มีการตีพิมพ์เผยแพร่อัลกอริทึมนี้ได้ไม่นาน NIST ก็ประกาศตามมาว่าอัลกอริทึมจำเป็นต้องได้รับการแก้ไขเพิ่มเติมเล็กน้อยเพื่อให้สามารถใช้งานได้เหมาะสม SHA สร้างไคเจสต์ที่มีขนาด 160 บิต

อัลกอริทึม SHA-1

SHA-1 เป็นอัลกอริทึมที่แก้ไขเพิ่มเติมเล็กน้อยจาก SHA การแก้ไขเพิ่มเติมนี้เป็นที่เชื่อกันว่าทำให้ อัลกอริทึม SHA-1 มีความปลอดภัยที่สูงขึ้น SHA-1 สร้างไคเจสต์ที่มีขนาด 160 บิต

อัลกอริทึม SHA-256, SHA-384 และ SHA-512

NIST เป็นผู้นำเสนออัลกอริทึมทั้งสามนี้ในปี 2001 เพื่อใช้งานร่วมกับอัลกอริทึม AES (ซึ่งเป็น อัลกอริทึมในการเข้ารหัสแบบสมมาตร) อัลกอริทึมเหล่านี้สร้างไคเจสต์ที่มีขนาด 256, 384 และ 512 บิต ตามลำดับ

นอกจากอัลกอริทึมสำหรับการสร้างไคเจสต์ที่กล่าวถึงไปแล้วนั้น อัลกอริทึมสำหรับการเข้ารหัสแบบสมมาตร เช่น DES สามารถใช้ในการสร้างไคเจสต์เช่นกัน วิธีการใช้งานอัลกอริทึมแบบสมมาตรเพื่อสร้างไคเจสต์คือ ให้เลือกกุญแจลับสำหรับการเข้ารหัสขึ้นมา 1 กุญแจโดยวิธีการเลือกแบบสุ่ม และต่อมาใช้กุญแจนี้เพื่อเข้ารหัสข้อความตั้งต้น แล้วใช้เฉพาะบล็อกสุดท้ายที่เข้ารหัสแล้วเพื่อเป็นไคเจสต์ของข้อความทั้งหมด (ไม่รวมบล็อกอื่นๆ ที่เข้ารหัสแล้ว) อัลกอริทึมแบบสมมาตรสามารถสร้างไคเจสต์ที่มีคุณภาพดี แต่ข้อเสียคือต้องใช้เวลาในการคำนวณไคเจสต์มาก

ไคเจสต์เป็นเครื่องมือที่สำคัญที่สามารถใช้ในการตรวจสอบว่าไฟล์ในระบบที่ใช้งานมีการเปลี่ยนแปลงแก้ไขหรือไม่ (ไม่ว่าจะโดยเจตนาหรือไม่ก็ตาม) บางครั้งการเปลี่ยนแปลงแก้ไขอาจถูกกระทำโดยผู้ที่ไม่มีสิทธิ์ เช่น ผู้บุกรุก เป็นต้น

วิธีการใช้ไคเจสต์เพื่อตรวจสอบไฟล์ในระบบคือให้เลือกใช้อัลกอริทึมหนึ่ง เช่น MD5 เพื่อสร้างไคเจสต์ของไฟล์ในระบบและเก็บไคเจสต์นั้นไว้อีกที่หนึ่งนอกระบบ ภายหลังจากระยะเวลาหนึ่ง

ที่กำหนดไว้ เช่น 1 เดือน ก็มาคำนวณไจเรสต์ของไฟล์เดิมอีกครั้งหนึ่ง แล้วเปรียบเทียบไจเรสต์ใหม่นี้กับไจเรสต์ที่เก็บไว้บนระบบว่าตรงกันหรือไม่ ถ้าตรงกัน ก็แสดงว่าไฟล์ในระบบยังเป็นปกติเช่นเดิม

ไจเรสต์ยังเป็นส่วนหนึ่งของการลงลายมือชื่ออิเล็กทรอนิกส์ กล่าวคือการลงลายมือชื่ออิเล็กทรอนิกส์ในปัจจุบันจะทำการลงลายมือชื่อกับไจเรสต์ของข้อความตั้งต้นแทนการลงลายมือชื่อกับข้อความตั้งต้นทั้งข้อความ

การเข้ารหัส XML สามารถแบ่งได้เป็น 3 ส่วนคือ

1. การเข้ารหัส XML แบบทั้งเอกสาร
2. การเข้ารหัสเฉพาะ element ในเอกสาร
3. การเข้ารหัสเฉพาะ ข้อมูลในเอกสาร

การเข้ารหัส XML นี้ทำให้สามารถจัดการได้ว่าข้อมูลไหนสมควรเป็นความลับข้อมูลไหน ควรจัดการอย่างไร มีความยืดหยุ่นสูง และช่วยลดเวลาในการเข้ารหัส

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <EncryptedData xmlns="http://www.w3.org/2001/04/xmldsig#" Id="Test"
  Type="http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml">
4   <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmldsig#tripledes-cbc" />
5   <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
6     <KeyName>theKey</KeyName>
7   </ds:KeyInfo>
8   <CipherData>
9     <CipherValue>v0oy2cty7WLj5RDNbVZo3UG/BtTt08IWWXf0Y/EHu/qivSa/p7wXP03mSXvtaGY87+
      hVjCrrH4mp
10    cKr+0xrTJumhVKHyptnHTpJksHxehqM9tC2yQDR0rZeLo76KrU2/FDS22DS3SLYDda3PUf5v0paU
11    k2TdoCJittXPwvbwULpZvP8/SBYVT+t8bqm24YEC9Q8shlmAc/OdXV03zfWIMKoqAZj1je03aImR
12    BobTnqd1o+iilmFUEV6DmgOpdwDuu+CsN9J3wE3C4co6Ish6zEqLFWmJAZwnytGTi6/t24fzRHeK
13    7f+SOIC+7yLjnnHPBfpvpZciC/IEg5jHd2qg4vhSKN0k6qfjUJpc5M4kl4dE79lv7LpF5m22tf8
14    z1pDXmwgk1YVl3ksIPXmHALi/y4ggx4uZpCAFR178k4bxiRL4NpoJx/ykl8JfidBrGvKluYB
15    nbdNCn/BjkzRUyyTfcgHQXfCeUe7JrueiQJjiIPwI3Ye8u4fipDfegCnbv/rb5/a7vFe2Z/RfRq
16    xUPBJQ9K0uP4+DMTP3qAfKL8F0ggQ5UnvnbLFYNxnx4D</CipherValue>
17   </CipherData>
18 </EncryptedData>
19
20

```

รูปที่ 3.9 แสดงตัวอย่าง SOAP Message มีการเข้ารหัสทั้งเอกสารแบบ triple-DES

3.5. การเพิ่มความปลอดภัยในการขนส่งข้อมูล (Transport – level Security)

Transport – level Security นิยมใช้ในการเพิ่มความปลอดภัยการสื่อสารของ เว็บเซอร์วิส ผ่าน protocol HTTP เช่น Secure Sockets Layer Transport หรือ Layer Security โดย HTTPS จะทำการตกลงระหว่าง Client กับ Server ที่ถูกรับรองด้วย Certificate ซึ่งเป็น Certificate ที่เขียนขึ้นมาเอง หรือ

Certificate ที่มีผู้รับรอง ก็ได้ ทำให้สามารถกำหนดได้ว่าจะทำการจัดการกับ security ใน message ได้อย่างอิสระอย่างไรก็ตาม โดย HTTPS จะทำการเข้ารหัส HTTP Message ที่มีการแลกเปลี่ยนกัน ไม่สามารถจัดการ security ได้หมด ยังมีบางส่วนเช่นใน SSL ไม่สามารถจัดการ JMS ได้

ในการใช้ Transport – level Security นิยมใช้ผ่าน HTTPS ซึ่งสามารถใช้ได้โดยแทน https:// ใน port address

3.5.1. หลักการทำงานของ SSL

Secure Sockets Layer (SSL) ใช้ในโพรโทคอลระดับแอปพลิเคชันคือ Hypertext Transfer Protocol (HTTP) ซึ่งเป็นการสื่อสารผ่านเว็บให้ปลอดภัย SSL ทำให้เกิดการสื่อสารอย่างปลอดภัยระหว่างไคลเอนต์และเซิร์ฟเวอร์ โดยการอนุญาตให้มีกระบวนการพิสูจน์ตัวตนกับการใช้งานลายเซ็นดิจิทัลสำหรับการรักษาความถูกต้องของข้อมูลและการเข้ารหัสข้อมูลเพื่อป้องกันความเป็นส่วนตัวระหว่างการสื่อสารข้อมูล

SSL นั้นได้รับการยอมรับอย่างกว้างขวางบน World Wide Web ในการใช้สำหรับตรวจสอบและเข้ารหัสลับการติดต่อสื่อสารระหว่าง client และ server หน้าที่ของ SSL จะแบ่งออกเป็น 3 ส่วนใหญ่ๆคือ

1. การตรวจสอบ server ว่าเป็นตัวจริง: ตัวโปรแกรม client ที่มีขีดความสามารถในการสื่อสารแบบ SSL จะสามารถตรวจสอบเครื่อง server ที่ตนกำลังจะไปเชื่อมต่อได้ว่า server นั้นเป็น server ตัวจริงหรือไม่ โดยใช้เทคนิคการเข้ารหัสแบบ public key ในการตรวจสอบใบรับรอง (certificate) และ public ID ของ server นั้น (โดยที่มียอดการที่ client เชื่อถือเป็นผู้ออกใบรับรองและ public ID ให้แก่ server นั้น)

หน้าที่นี้ของ SSL เป็นหน้าที่ที่สำคัญ โดยเฉพาะอย่างยิ่งในกรณีที่ client ต้องการที่จะส่งข้อมูลที่เป็นความลับ (เช่น หมายเลข credit card) ให้กับ server ซึ่ง client จะต้องตรวจสอบก่อนว่า server เป็นตัวจริงหรือไม่

2. การตรวจสอบว่า client เป็นตัวจริง: server ที่มีขีดความสามารถในการสื่อสารแบบ SSL จะใช้เทคนิคเช่นเดียวกับในหัวข้อที่แล้วในการตรวจสอบ client หรือผู้ใช้ว่าเป็นตัวจริงหรือไม่ โดยจะตรวจสอบใบรับรองและ public ID (ที่มียอดการที่ server เชื่อถือเป็นผู้ออกให้) ของ client หรือผู้ใช้นั้น หน้าที่นี้ของ SSL จะมีประโยชน์ในกรณีเช่น ธนาคารต้องการที่จะส่งข้อมูลลับทางการเงินให้แก่ลูกค้าของตนผ่านทางเครือข่าย Internet (server ก็จะต้องตรวจสอบ client ก่อนว่าเป็น client นั้นจริง)

3. การเข้ารหัสลับการเชื่อมต่อ: ในกรณีนี้ ข้อมูลทั้งหมดที่ถูกส่งระหว่าง client และ server จะถูกเข้ารหัสลับ โดยโปรแกรมที่ส่งข้อมูลเป็นผู้เข้ารหัสและโปรแกรมที่รับข้อมูลเป็นผู้ถอดรหัส (โดยใช้

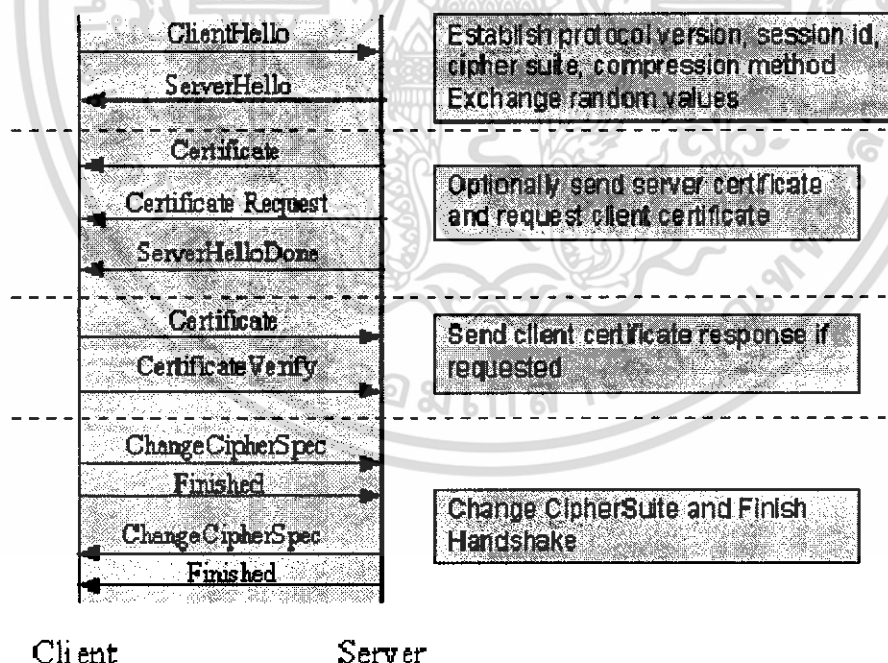
วิธี public key) นอกจากการเข้ารหัสลับในลักษณะนี้แล้ว SSL ยังสามารถปกป้องความถูกต้องสมบูรณ์ของข้อมูลได้อีกด้วย กล่าวคือ ตัวโปรแกรมรับข้อมูลจะทราบได้หากข้อมูลถูกเปลี่ยนแปลงไปในขณะกำลังเดินทางจากผู้ส่งไปยังผู้รับ

โพรโทคอล SSL อนุญาตให้สามารถเลือกวิธีการในการเข้ารหัส วิธีสร้างไคเจสต์ [*1] และลายเซ็นดิจิทัล ได้อย่างอิสระก่อนการสื่อสารจะเริ่มขึ้นขึ้น ตามความต้องการของทั้งเว็บเซิร์ฟเวอร์และบราวเซอร์ ทั้งนี้เพื่อเพิ่มความยืดหยุ่นในการใช้งาน เปิดโอกาสให้ทดลองใช้วิธีการในการเข้ารหัสวิธีใหม่ รวมถึงลดปัญหาการส่งออกวิธีการเข้ารหัสไปประเทศที่ไม่อนุญาต

กระบวนการในการเริ่มต้นการสื่อสารผ่านชั้น SSL แบ่งเป็น 4 ขั้นตอนคือ

1. ประกาศชุดวิธีการเข้ารหัส ไคเจสต์ และลายเซ็นดิจิทัลที่สนับสนุนของทั้งไคลเอ็นต์และเซิร์ฟเวอร์
2. การพิสูจน์ตัวตนของเซิร์ฟเวอร์ต่อไคลเอ็นต์
3. การพิสูจน์ตัวตนของไคลเอ็นต์ต่อเซิร์ฟเวอร์ ถ้าจำเป็น
4. ไคลเอ็นต์และเซิร์ฟเวอร์ตกลงชุดวิธีการเข้ารหัส การสร้างไคเจสต์ และการใช้ลายเซ็นดิจิทัล

ตามรูป



รูปที่ 3.10 กระบวนการเริ่มต้นการติดต่อสื่อสารของโพรโทคอล SSL

ขั้นตอน 1: ประกาศชุดวิธีการเข้ารหัส โดเจสต์ และลายเซ็นดิจิทัลที่สนับสนุนของทั้งไคลเอนต์และเซิร์ฟเวอร์

ไคลเอนต์และเซิร์ฟเวอร์ส่งข้อความเริ่มต้นการสื่อสาร (Hello message) ซึ่งประกอบไปด้วยเวอร์ชันของโปรโตคอลที่ใช้ วิธีการเข้ารหัสที่เว็บเซิร์ฟเวอร์และไคลเอนต์สนับสนุน หมายเลขระบุการสื่อสาร (Session identifier) รวมถึงวิธีการบีบอัดข้อมูลในการสื่อสารที่สนับสนุน

หมายเลขระบุการสื่อสารที่เกิดขึ้น ใช้สำหรับตรวจสอบการเชื่อมต่อระหว่างไคลเอนต์และเซิร์ฟเวอร์ ถ้ามีการเชื่อมต่อก่อนหน้านี้เกิดขึ้น แสดงว่าได้มีการตกลงวิธีการสื่อสารแล้ว สามารถเริ่มต้นส่งข้อมูลได้ทันที เป็นการลดเวลาติดต่อสื่อสารลง

ขั้นตอน 2: การพิสูจน์ตัวตนของเซิร์ฟเวอร์ต่อไคลเอนต์

ถัดมาเว็บเซิร์ฟเวอร์ทำการส่ง Certificate หรือใบยืนยันความมีตัวตนของเซิร์ฟเวอร์ ไคลเอนต์จะทำการตรวจสอบ Certificate กับผู้ให้บริการ Certificate Authority ที่ได้ตั้งค่าไว้ เพื่อยืนยันความถูกต้องของ Certificate ของเซิร์ฟเวอร์

ขั้นตอน 3: การพิสูจน์ตัวตนของไคลเอนต์ต่อเซิร์ฟเวอร์ ถ้าจำเป็น

เซิร์ฟเวอร์สามารถร้องขอ Certificate จากไคลเอนต์เพื่อตรวจสอบความถูกต้องของ Client ด้วยก็ได้ ใช้ในกรณีที่มีการจำกัดการใช้งานเฉพาะไคลเอนต์ที่ต้องการเท่านั้น ซึ่ง SSL สนับสนุนการตรวจสอบได้จากทั้งเซิร์ฟเวอร์และไคลเอนต์ ขึ้นอยู่กับการเลือกใช้งานในขณะติดต่อสื่อสารที่เกิดขึ้นนั้น

ขั้นตอน 4: ไคลเอนต์และเซิร์ฟเวอร์ตกลงชุดวิธีการเข้ารหัส การสร้างโดเจสต์ และการใช้ลายเซ็นดิจิทัล

ขั้นตอนการตรวจสอบ Certificate ที่เซิร์ฟเวอร์ร้องขอจากไคลเอนต์จะมีหรือไม่มีก็ได้ ขึ้นอยู่กับการตั้งค่าบนเซิร์ฟเวอร์ หลังจากขั้นตอนการตรวจสอบเสร็จสิ้น เซิร์ฟเวอร์และไคลเอนต์จะตกลงการใช้งานวิธีการเข้ารหัสระหว่างกันโดยใช้ค่าที่ได้จากการประกาศในขั้นตอนแรก

วิธีการแลกเปลี่ยนกุญแจในการเข้ารหัส (Key exchange method) คือการกำหนดกลไกการแลกเปลี่ยนกุญแจที่ใช้ในการเข้ารหัสระหว่างการสื่อสาร โดยทั้งไคลเอนต์และเซิร์ฟเวอร์จะใช้กุญแจนี้ในการเข้ารหัสและถอดรหัสข้อมูล ใน SSL เวอร์ชัน 2.0 จะสนับสนุนวิธีการแลกเปลี่ยนกุญแจแบบ RSA ส่วน SSL เวอร์ชัน 3.0 ขึ้นไปจะสนับสนุนวิธีการอื่นๆ เพิ่มเติมเช่นการใช้ RSA ร่วมกับการใช้ Certificate หรือ Diffie-Hellman เป็นต้น

วิธีการเข้ารหัสในปัจจุบันแบ่งเป็นสองวิธีคือ การใช้กุญแจเดียวกันในการเข้ารหัสและถอดรหัส อาจเรียกกุญแจนี้ว่า Session key หรือ Secret key ส่วนอีกวิธีการคือ การใช้กุญแจคนละตัวในการเข้ารหัสและถอดรหัส ประกอบไปด้วยกุญแจสาธารณะและกุญแจส่วนตัวซึ่งเป็นที่กันเสมอ การเข้ารหัสด้วยกุญแจใด จะต้องถอดรหัสด้วยกุญแจที่คู่กันและตรงกันข้ามเท่านั้น มักใช้วิธีการเข้ารหัสด้วยกุญแจคนละตัวมาใช้ในการเข้ารหัส Session key และส่งไปให้ฝั่งตรงข้ามก่อนการสื่อสารจะเกิดขึ้นรวมเรียกว่าวิธีการแลกเปลี่ยนกุญแจในการเข้ารหัส

SSL ใช้วิธีการเข้ารหัสด้วยกุญแจสมมาตร หรือกุญแจเดียวในการเข้ารหัสและถอดรหัส ตามที่กล่าวข้างต้น วิธีการเข้ารหัสคือ การเข้ารหัสด้วย DES และ 3DES (Data Encryption Standard), วิธีการเข้ารหัสด้วย IDEA ส่วน RC2 และ RC4 เป็นวิธีการเข้ารหัสของ RSA รวมถึงวิธีการเข้ารหัสแบบ Fortezza สำหรับความยาวของการเข้ารหัสที่ใช้คือ 40 บิต, 96 บิต และ 128 บิต

การสร้าง Message Authentication Code (MAC) เพื่อใช้สำหรับการยืนยันความถูกต้องของข้อมูลระหว่างการสื่อสารและป้องกันการปลอมข้อมูล ส่วนฟังก์ชันสร้างโคแฮชต์ที่ SSL สนับสนุนและเลือกใช้ได้ในปัจจุบันคือ MD5 ขนาด 128 บิต และ SHA-1 (Secure Hash Algorithm) ขนาด 160 บิต ซึ่งจะได้วิธีการที่ทั้งสองฝ่ายสนับสนุนและเหมาะสมซึ่งเป็นขั้นตอนสุดท้ายก่อนการสื่อสารที่มีการเข้ารหัสจะเริ่มต้นขึ้น

3.6. รูปแบบของคีย์และการนำไปใช้

จากการศึกษาเกี่ยวกับความปลอดภัยของเว็บเซอร์วิสทั้งจาวาและดอทเน็ตพบว่า “คีย์” หรือใบรับรองอิเล็กทรอนิกส์ (Certificates) มีความสำคัญมากในทุกๆส่วนซึ่งการสร้างและการนำไปใช้ของคีย์แต่ละชนิดก็มีความแตกต่างกันอย่างเห็นได้ชัดดังอธิบายด้านล่างนี้

ตารางที่ 3.1 แสดง การเปรียบเทียบความสามารถของคีย์แต่ละตัว

Types of Certificate							
Extension	Type	Holds Multiple Certificates?	Holds Private Keys?	Password Required?	Microsoft	Java	Notes
*.p7b	PKCS#7	Yes	No	No	Yes	Yes	IE binary public key export. Can optionally

							contain multiple certs, e.g. a certificate chain. IBM's Keyman will create and manage this format of keyring.
n/a	PKCS#8	No	Yes	Yes	n/a	n/a	The private-key information syntax standard. This defines a method to store Private Key Information.
*.csr,.p10	PKCS#10	No	No	No	n/a	n/a	The certification request syntax standard. This describes a syntax for certification requests. Contains the public key, signed with the private key.
n/a	PKCS#11 / CryptoKi	Yes	Yes	Yes	n/a	n/a	The cryptographic token interface standard. This defines a technology independent programming interface for cryptographic devices such as smartcards.
.p12, .pfx	PKCS#12	Yes	Yes	Yes	Yes	Yes	The personal information exchange syntax standard. This describes a portable format

							for storage and transportation of user private keys, certificates etc. Where a developer stores his code signing keys. .keystore is not in this format.
*.pem	PEM	n/a	No	n/a	n/a	n/a	Privacy Encoded Mail format for sending certs embedded in email, typically SSL cert.
*.sst	n/a	n/a	Yes	n/a	Yes	n/a	Windows certificate store.
*.stl	n/a	n/a	No	n/a	Yes	n/a	Windows certificate trust list. Used my cryptext.dll.
*.usr	X.509/DER binary format	n/a	No	n/a	Yes	n/a	user certificates.
*.asc	PGP	n/a	No	n/a	n/a	n/a	Pretty Good Privacy.
*.ca	X.509/DER binary format	No	No	No	Yes	Yes	root certificates.
*.cer	X.509/DER binary format	No	No	No	Yes	Yes	Sun Java 1.3+ user certs.
*.cer	X.509/DER BASE64 encoded	No	No	No	Yes	Yes	Sun Java 1.3+ user certs.
*.crt	X.509/DER binary format	No	No	No	Yes	Yes	Thawte root certificates, Sun Java 1.3+ cacerts.

*.db	proprietary binary?	Yes	Yes	No	Yes	n/a	Netscape export of the entire set of keys. Contains multiple certs with private keys.
.keystore, cacerts.	JKS	Yes	Yes	Yes	No	Yes	Sun's keyring format. Can optionally include private key, authentication chain and friendly name. Sun never imports/exports the private key, though .keystore contains it.

3.7. Certificates

สำหรับค่า Public key encryption เป็นระบบที่มีความน่าเชื่อถือและจะมีกลไกในการยืนยันและกระจายค่า Public keys ซึ่งค่าที่ส่งนี้ Windows Server 2003 รองรับ PKI ใช้ Digital Certificates เพื่อยืนยันบุคคลนั้นอยู่ในองค์กร

การใช้ Public key จะต้องมีการบริหารที่เรียกว่า Certification Authority (CA) ซึ่ง CA นี้สามารถที่เป็นบริษัทที่แยกดูแลจากภายนอกก็ได้ โดยจะกำหนดเป็นชั้นๆของซอฟต์แวร์ในเครื่อง ซึ่งการเลือก CA เราต้องขึ้นอยู่กับองค์กรที่ติดต่อด้วย

การบรรจุการรับรองจาก CA สามารถทำได้ด้วยมือ และเราก็ส่งคำร้องขอที่แน่นอนไปที่ CA หรือใช้แบบอัตโนมัติ คือระบบจะทำงานเป็นแบกราวนด์ในการทำงานแบบปกติ ซึ่ง CA จะรับ Public key และ Private key ว่าตรงกันหรือไม่ ซึ่งค่าที่เป็น Private จะเข้ารหัสเก็บไว้ในเครื่องผู้ใช้ตามแบบที่ถูกเข้ารหัส และค่า Public key จะอยู่ใส่ส่วนของ Certificate หรือผู้รับรอง โดยค่า Public key นี้จะใช้ในการกระจายให้กับบุคคลอื่นๆที่ต้องการ

3.7.1. ส่วนประกอบของ Certificate

Certificate คือ key ที่ใช้ในระบบ security ซึ่งมีส่วนประกอบดังนี้
Public key

เป็นตัวเลขที่เกี่ยวข้องกันซึ่งเป็นที่รู้กันว่ามีค่าน่าเชื่อถือในการติดต่อกันระหว่าง entity นิยมใช้คล้ายกับการ verify signature

Digitally Signed

ถ้ามีการใช้ digitally signed กับ entity ใดและมี signature ที่พิสูจน์ข้อมูลนั้น การนำข้อมูลนั้นไปใช้ จะต้องมีการใช้ private key เพื่อถอด

Identity

ที่อยู่แน่ชัดของ entity ในบางครั้ง entity ก็คือ public key นั่นเอง

Signature

ข้อมูลที่ถูกเข้ารหัสด้วย private key

Private key

เป็นชุดของตัวเลขที่รู้เฉพาะผู้ที่เป็นเจ้าของ โดยจะมากคู่กับ public key เสมอ

Entity

คือบุคคล , องค์กร , โปรแกรมคอมพิวเตอร์ , องค์กรธุรกิจ , ธนาคาร หรือสิ่งอื่นๆที่คุณสามารถอ้างถึงได้

3.7.2. การใช้ CA ภายในหรือภายนอก

เราสามารถวางระบบทั้งการใช้ CA ทั้งภายใน และภายนอก ซึ่งถ้าใช้ภายในเราต้องตั้ง PKI ในเครือข่าย หรือจะใช้ CA ของค่ายอื่นก็ได้ หรือใช้ทั้งคู่ ซึ่งการกำหนดจะต้องตรวจสอบกลุ่มที่ติดต่อว่าอยู่ในบริษัทเดียว หรือต่างบริษัท

ถ้าเราต้องการให้มีความปลอดภัยในองค์กรก็ตั้ง CA ขึ้นโดยใช้ Certificate Services ใน Windows Server 2003 ทำเป็น CA และให้ผู้ใช้ทั้งหมดทำการติดต่อใน CA ภายในนี้ การติดต่อข้ามที่อื่นทำไม่ได้เนื่องจากต้องการตรวจสอบ และระบุจาก CA

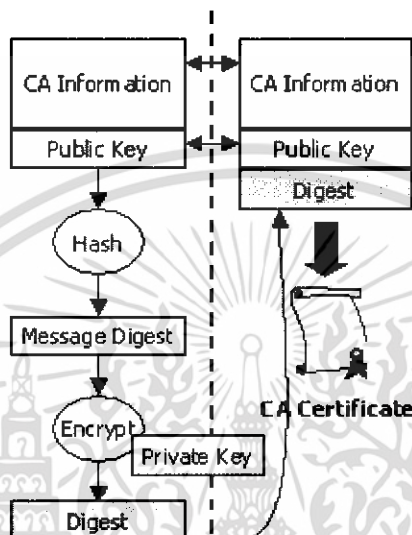
สำหรับการติดต่อภายนอกเราสามารถที่จะใช้ Commercial CA ที่ติดต่อเช่น Thawe และ VeriSign Inc

3.7.3. Certification Authority (CA)

คือหน่วยผู้ประกอบการรับรอง ที่เป็นหน่วยงานภายนอก (Trusted Third Party) ซึ่งมีหน้าที่หลักคือ เป็นผู้ออกใบรับรองอิเล็กทรอนิกส์ให้กับบุคคลหรือสิ่งอื่นๆ (ในที่นี้จะเรียกรวมว่าเป็น Subject) ทั้งนี้ Self-signed Certificate จะเป็นสิ่งที่ระบุตัวตนของหน่วยผู้ประกอบการรับรองเอง ดังนั้น หน่วยผู้ประกอบการรับรองจึงเป็นจุดเริ่มต้นของการสร้างความไว้วางใจ (Trust) แต่ละเอ็นทิตีจะต้องมอบความไว้วางใจต่อหน่วยผู้ประกอบการรับรองและสิ่งใดก็ตามที่หน่วยผู้ประกอบการรับรองได้ออกใบรับรองให้

3.7.4. การสร้าง Self-signed Certificate

ในการเริ่มต้นเป็นหน่วยผู้ประกอบการรับรอง จะต้องสร้างใบรับรองอิเล็กทรอนิกส์ของ CA เองก่อน ทั้งนี้ขั้นตอนการสร้าง Self-signed Certificate ของ CA เป็นดังในรูปที่ 3.11



รูปที่ 3.11 แสดงกระบวนการสร้าง Self-signed Certificate

หน่วยผู้ประกอบการรับรองเป็นหัวใจสำคัญของ PKI และรับผิดชอบหน้าที่สำคัญหลายประการ ได้แก่ Certificate Management กล่าวคือ Issuance, Revocation, Update และ Renewal การเผยแพร่ใบรับรองอิเล็กทรอนิกส์และ CRL (Certificate Revocation List) และทำการบันทึกที่ถือออกจากเหตุการณ์ต่างๆ ที่เกิดขึ้นในระบบ ขั้นตอนโดยทั่วไปในการออกใบรับรองให้กับ Subject ใดๆ นั้นจะประกอบไปด้วย

1. Subject ส่งคำร้องขอ (Certificate Request)
2. Subject เป็นผู้สร้างกุญแจคู่โดยหน่วยผู้ประกอบการรับรองเป็นผู้จัดเตรียมฟังก์ชันนี้ให้
3. หน่วยผู้ประกอบการรับรองจะจัดการออกใบรับรองให้ ทั้งนี้ คำร้องดังกล่าว อาจเป็นไปได้ตั้งแต่ขอใหม่ การปรับเปลี่ยนหรือขอให้ออกให้ใหม่
4. รอการอนุมัติจากหน่วยผู้ประกอบการรับรอง
5. หน่วยผู้ประกอบการรับรองตรวจสอบ Identity ของผู้ร้องขอ เมื่อผ่านแล้ว หน่วยผู้ประกอบการรับรองจะทำการ Sign Request นั้นโดยการสร้างใบรับรองอิเล็กทรอนิกส์
6. เผยแพร่ใบรับรองอิเล็กทรอนิกส์ดังกล่าว เพื่อให้ผู้ร้องขอมารับไป

3.7.5. Registration Authority (RA)

คือ หน่วยรับผิดชอบในการดำเนินงานจริง หน่วยผู้ประกอบการรับรองบางแห่งอาจจะแบ่งหน้าที่บางอย่างให้กับ RA ใน RFC 2510 Internet Public Key Infrastructure Certificate Management Protocols อ้างถึงหน้าที่ของ RA ไว้ว่า คือ การตรวจสอบความถูกต้องของบุคคล (Personal Authentication) การแจกจ่าย Token (Token distribution) รายงานการเพิกถอน ใบรับรอง (Revocation reporting) การกำหนดชื่อ (Name assignment) การสร้างกุญแจคู่ (Key generation) การจัดเก็บกุญแจคู่ (Archival of key pairs) เป็นต้น ในทางปฏิบัติแล้ว ส่วนใหญ่ RA จะทำหน้าที่ตรวจสอบความถูกต้องของบุคคลหรือ Subject ใดๆ ระหว่างกระบวนการออกใบรับรองอิเล็กทรอนิกส์ (Certificate Enrollment Process)

3.7.6. Certificate Management Protocol (CMP)

คือ ขั้นตอนหรือโพรโตคอลในการจัดการใบรับรองอิเล็กทรอนิกส์ โดยที่ Certificate Management Protocol หลายแบบได้ระบุถึงขั้นตอน Certificate Enrollment ในบางกรณีก็ระบุถึงขั้นตอน Certificate Revocation Process เวนเดอร์บางแห่งก็เพิ่มเติมฟังก์ชันการทำงานของ โพรโตคอล นอกเหนือจากนี้เช่น Key recovery และ Automated certificate renewal Public Key Cryptographic Standard (PKCS) เป็นชุดโพรโตคอลมาตรฐานของ RSA Security ที่ระบุวิธีการรักษาความปลอดภัยของการแลกเปลี่ยนข้อมูลระหว่างกัน PKCS #1, #3, #5, #6, #7, #8, #9, #10, #11, #12 และ #15 เป็นมาตรฐานที่มีการใช้งานในปัจจุบัน ส่วน PKCS #13 และ #14 ยังคงอยู่ในระหว่างการอนุมัติ ส่วน PKCS #2 และ #4 ได้ถูกรวมเข้ากับ PKCS #1 แล้ว

PKCS #1 : RSA Cryptography Standard

PKCS #2 : ถูกรวมเข้ากับ PKCS #1 แล้ว

PKCS #3 : Diffie-Hellman Key Agreement Standard

PKCS #4 : ถูกรวมเข้ากับ PKCS #1 แล้ว

PKCS #5 : Password-Based Cryptography Standard

PKCS #6 : Extended-Certificate Syntax Standard

PKCS #7 : Cryptographic Message Syntax Standard

PKCS #8 : Private-Key Information Syntax Standard

PKCS #9 : Selected Attribute Types

PKCS #10 : Certification Request Syntax Standard

PKCS #11 : Cryptographic Token Interface Standard

PKCS #12 : Personal Information Exchange Syntax Standard

PKCS #13 : Elliptic Curve Cryptography Standard

PKCS #15 : Cryptographic Token Information Format Standard

ในปี 1999 IETF ได้อนุมัติ RFC 2510 Internet Public Key Infrastructure Certificate Management Protocols และ RFC 2511 Internet Public Key Infrastructure Certificate Request Management Framework (CRMF) จากนั้น PKIX Working Group ได้พัฒนา Certificate Management Protocol (CMP) บนพื้นฐานจากมาตรฐานเหล่านี้ โดย CMP เป็นอีกทางเลือกหนึ่งนอกเหนือไปจาก PKCS สำหรับจัดการ Certificate Request และ Response Message ทั้งนี้ CMP รองรับฟังก์ชันการทำงานของจัดการใบรับรอง หลากหลายกว่า เช่น Cross-Certification Request และ Response Message, Revocation Request และ Response Message, Certificate and CRL Distribution Message แต่ CMP เป็นโพรโตคอลที่ค่อนข้างซับซ้อน ที่แม้ซอฟต์แวร์ CA จะรองรับแต่ โปรแกรมต่างๆ หรือ อุปกรณ์ก็ยังไม่รองรับเป็นที่แพร่หลายนัก

3.7.7. Certificate Revocation

คือกระบวนการเพิกถอนใบรับรองอิเล็กทรอนิกส์ (Certificate Revocation) เนื่องจากหมดอายุ หรือใบรับรองอิเล็กทรอนิกส์ถูกผู้อื่นนำไปใช้ วิธีที่นิยมใช้ ได้แก่

CRL (Certificate Revocation List) เป็นรายชื่อของใบรับรองที่ถูกเพิกถอน ใน RFC 2459 Internet X.509 Public Key Infrastructure Certificate and CRL Profile ได้ระบุรูปแบบของ X.509 CRL Version 2 ไว้

OCSP (Online Certificate Status Protocol) ในการใช้งาน PKI กับงานบางประเภท ช่วงเวลาระหว่างการอัปเดต CRL แต่ละครั้งไม่เป็นที่ยอมรับ ดังนั้นจึงเกิด OCSP ขึ้น ซึ่งเป็นกลไกในการตรวจสอบการเพิกถอนของใบรับรองอิเล็กทรอนิกส์แบบเรียลไทม์ เ็นที่ติจะส่ง Certificate Status Check ไปยัง OCSP Responder ที่อาจเป็นหน่วยผู้ประกอบการรับรองหรือ แบ่งหน้าที่ให้กับหน่วยอื่นก็ได้ ทั้งนี้ใบรับรองอิเล็กทรอนิกส์จะไม่ได้รับการยอมรับจนกว่า OCSP Responder จะตอบ Message มา ยืนยันหรือปฏิเสธสถานะของใบรับรองอิเล็กทรอนิกส์นั้น

3.7.8. Certificate Repository

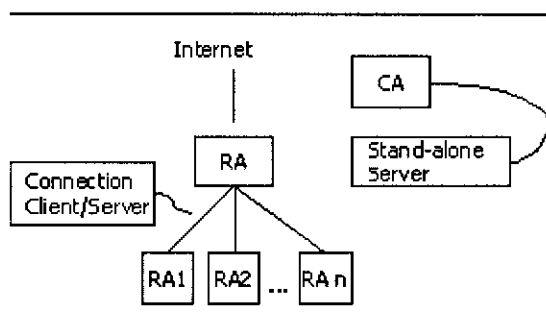
ใช้สำหรับจัดเก็บและประกาศใบรับรองอิเล็กทรอนิกส์ให้สาธารณะรับทราบ ในการพัฒนาระบบเล็กๆ ส่วนนี้อาจจะไม่จำเป็น เพราะอาจทำการแลกเปลี่ยนใบรับรองหรือ CRL ผ่านทางจดหมายอิเล็กทรอนิกส์ได้ แต่เมื่อต้องเกี่ยวข้องกับใบรับรองจำนวนมาก ก็จำเป็นต้องมีการจัดเก็บให้เป็นไปในทิศทางเดียวกัน ซึ่งอาจจะเลือกใช้ Directory Services หรือ FTP และ HTTP ก็ได้

3.7.9. Time Stamp Authority (TSAs)

เนื่องจากองค์ประกอบเรื่องความปลอดภัยข้อหนึ่งคือ Non-repudiation (ความแน่ใจว่าคู่สื่อสารไม่สามารถที่จะปฏิเสธการกระทำทางอิเล็กทรอนิกส์ที่เกิดขึ้นได้) เรื่องของเวลาจึงเป็นส่วนประกอบสำคัญส่วนหนึ่ง ยกตัวอย่างการสื่อสารผ่านจดหมายอิเล็กทรอนิกส์ หากมีการใช้ Time Stamp Protocol (TSP) เช่น หากนาย ก ต้องการส่งจดหมายอิเล็กทรอนิกส์ให้นาย ข โดยแนบลายมือชื่อดิจิทัลไปด้วย นาย ก จะส่งไปยัง Time Stamping Authority (TSA) ก่อนเพื่อทำการ Sign Request ของนาย ก จากนั้น TSA ส่งกลับคืนนาย ก แล้วนาย ก จึงส่งจดหมายอิเล็กทรอนิกส์ให้นาย ข ทั้งนี้ หากนาย ข ปฏิเสธว่าไม่ได้รับจดหมาย นาย ก สามารถนำ Response จาก TSA มาอ้างอิงภายหลังได้ แต่ทั้งสองคนจะต้องให้ความไว้วางใจต่อ TSA ล่วงหน้าก่อนแล้ว

3.7.10. แผนโครงสร้างของ OpenCA

ส่วนที่เป็น CA จะแยกออกมาเป็นเซิร์ฟเวอร์ที่ไม่เชื่อมต่อกับเซิร์ฟเวอร์อื่นหรืออินเทอร์เน็ต เนื่องจากต้องการความปลอดภัยสูงเพราะเป็นเซิร์ฟเวอร์ที่ติดตั้ง CA Software และเก็บกุญแจส่วนตัวของ CA ไว้ด้วยเซิร์ฟเวอร์ RA จะมีความซับซ้อนมากกว่า กล่าวคือ จะมีเว็บเซิร์ฟเวอร์และเซิร์ฟเวอร์ LDAP รันอยู่ โดยอย่างแรกจะทำหน้าที่รองรับ Request จาก Subject ใดๆ ในการขอใบรับรองอิเล็กทรอนิกส์ ก่อนที่ RA จะเป็นผู้นำไปให้ CA รับรองต่อ ส่วนอย่างหลัง จะทำหน้าที่เป็นที่เก็บและเผยแพร่ใบรับรองอิเล็กทรอนิกส์ จากรูปที่ 3.12 จะพบว่า RA สามารถมีได้หลายแห่ง ทั้งนี้เพื่อแบ่งเบาภาระนั่นเอง

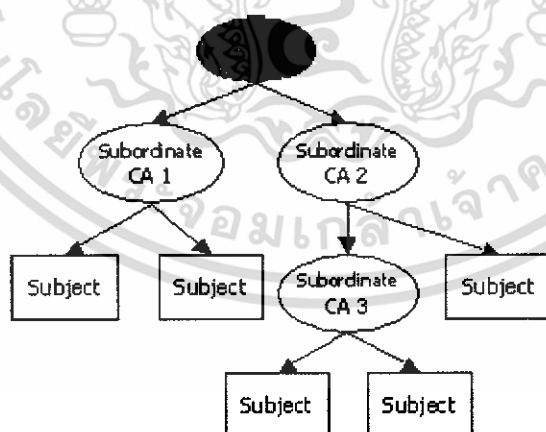


รูปที่ 3.12 แสดงโครงสร้างของ OpenCA

3.7.11. แผนโมเดลการให้ความเชื่อถือ (Trust Model)

Hierarchy

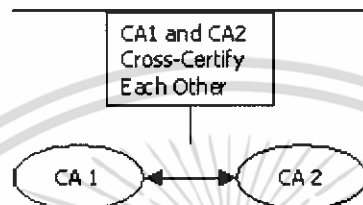
ในการสร้างหน่วยผู้ประกอบการรับรองนั้น หากเป็นการสร้างเพื่อใช้กันภายในองค์กรเองแล้ว CA นั้นก็ย่อมจะได้รับความไว้วางใจจากบุคลากรในองค์กร แต่หากบุคคลในองค์กรต้องการที่จะติดต่อสื่อสารกับบุคคลภายนอกโดยใช้ใบรับรองอิเล็กทรอนิกส์ที่ออกให้โดยองค์กรของตน บุคคลภายนอกอาจไม่รู้จักและไม่ให้ความไว้วางใจ CA ภายใน บางครั้ง การรับรองคู่กุญแจที่สร้างเองก็ไม่เป็นที่ยอมรับ จำเป็นต้องขอให้ CA อื่นที่เป็นรู้จักและได้รับความไว้วางใจมากกว่ารับรองให้ ดังในรูปที่ 3.13 Root CA จะเป็นผู้รับรองให้ Subordinate CA1 และ 2 ใครก็ตามที่มอบความไว้วางใจให้กับ Root CA ก็จะถือว่ามอบให้ Subordinate CA1 และ 2 ด้วยเช่นกัน เรียกว่าเป็น Trust Model แบบ Hierarchy (แบบลำดับ)



รูปที่ 3.13 แสดง Trust Model แบบ Hierarchy

Cross-Certification

โมเดลการมอบความไว้วางใจดังในรูปที่ 3.14 นั้นเรียกว่า Cross Certification หน่วยผู้ประกอบการรับรองจะผลิตกัน Sign ใบรับรองของ CA ให้แก่กัน ผู้ที่ได้ใบรับรองจากไม่ว่าจะ CA1 หรือ CA2 ก็จะมอบความไว้วางใจให้กับทั้ง 2 ที่ ทั้งนี้ เป็นเรื่องยากที่จะทำให้ CA แต่ละแห่งมอบความไว้วางใจให้แก่กัน จึงไม่เป็นที่นิยมนัก ผู้ใช้ที่ให้บริการจาก CA คนละแห่งจึงต้องมีหน้าที่เลือกมอบความไว้วางใจให้กับ CA ใดๆ เอง



รูปที่ 3.14 แสดง Trust Model แบบ Cross-Certification

โมเดลการมอบความไว้วางใจนั้น จะมีการออกแบบและใช้งานก็ต่อเมื่อมีหน่วยผู้ประกอบการรับรองมากกว่า 1 แห่งขึ้นไป ดังนั้น การทดสอบใช้งานซอฟต์แวร์ในบทความนี้เป็นารออกแบบหน่วยผู้ประกอบการรับรองที่ใช้งานภายในเพียงหน่วยเดียว จึงยังมิได้กำหนดโมเดลการมอบความไว้วางใจกันไว้

3.7.12. เข้าใจหน้าที่ของ PKI

ใน Windows Server 2003 PKI จะมีงานดังนี้

- Publish certificates สามารถที่จะสร้าง การรับรองบน Web Site หรือ Active Directory ที่ผู้ใช้เครื่อง แอปพลิเคชัน
- Enroll Clients ลงทะเบียนจากลูกค้าเข้ามาที่ CA โดยจะได้ชื่อลูกค้า
- User Certificates เมื่อลูกค้าร้องขอ และรับการรับรอง สามารถที่สื่อสารได้อย่างปลอดภัย และสามารถที่ทำหน้าที่กับรายการที่ออกให้
- Renew certificates เป็นการต่ออายุเมื่อครบ
- Revoke certificates เป็นการถอนค่าที่รับรองแล้วไปเพิ่มในใน Certificate revocation list (CRL)

3.8. การออกแบบ Public Key Infrastructure

ส่วนประกอบของเครือข่ายส่วนมากคือการวางระบบ Public key infrastructure ซึ่งต้องมีการวางระบบอย่างระมัดระวัง โดยมีขั้นตอนดังนี้

1. กำหนดความต้องการ Certificate
2. การสร้างโครงสร้างพื้นฐาน Certification
3. การกำหนดค่า Certification

3.8.1. กำหนดความต้องการ Certificate

ในเฟสนี้ต้องมีการออกแบบเครือข่าย และกำหนดลูกข่ายที่มีความจำเป็นที่ต้องใช้ และบอกถึงว่า Certificate ช่วยอะไรในการทำงาน ชนิดการรับรองที่ถูกขั้วจำเป็น รวมถึงการตอบคำถามเกี่ยวกับระบบความปลอดภัย และยุทธศาสตร์ในการดำเนินงานด้านความปลอดภัย PKI ใช้กับเครื่อง Windows Server 2003 สามารถที่สร้าง และรองรับสิ่งต่างๆดังนี้

- Digital Signatures รองรับลายเซ็นอิเล็กทรอนิกส์
- การเข้ารหัสระบบไฟล์ผู้ใช้ และการซ่อมคืนการรับรอง คือใช้กับ EFS ใน Windows Server 2003 และการกำหนด Recovery agents หรือกลุ่มที่สามารถที่จะนำไฟล์กลับคืนมาให้ได้ โดยกลไกทั้งหมดใช้ PKI ในการจัดการ
- Internet authentication สามารถที่ใช้สร้างการเชื่อมต่อที่ปลอดภัยในการติดต่อระหว่างผู้ Server กับลูกข่าย
- IP Security หรือ IPSec สามารถที่ใช้กลไกนี้ในการเข้ารหัสลับ และใช้กับการป้องกันข้อมูลในการโอนถ่าย
- Secure e-mail เข้ารหัสอิเล็กทรอนิกส์
- Smart Card logon เป็นการใช้กับวงจรด้านการรับรองด้วยบัตร ด้วยการเก็บ Smart card เป็น Private key เพื่อเข้าใช้เครือข่าย
- Software code signing เป็นการใช้กับ Microsoft's Authenticode technology ที่ใช้ในการดาวน์โหลด และติดตั้งโดยตรงจากผู้ประกาศโดยไม่มีการแก้ไข
- Wireless network authentication เพิ่มความปลอดภัยใน Wireless LAN ที่อาจมีการแอบดักฟังได้สิ่งที่จะนำแอปพลิเคชันใดมาใช้ต้องพิจารณาถึงความต้องการความปลอดภัยแค่ไหนในกลุ่มคนใด ซึ่งเราอาจจะต้องพิจารณาร่วมกับการทำงานเช่น
- ถ้ามีการต่ออินเทอร์เน็ตจึงใช้ซอฟต์แวร์ที่เกี่ยวข้องกับอินเทอร์เน็ตได้

- ความปลอดภัยแบ่งเป็น 4 ระดับคือ พื้นฐาน กลาง สูง และภายนอก
- ทั่วไปองค์กรเราต้องการใช้งานด้าน e-mail, EFS
- ป้องกันข้อมูลระดับกลาง เช่นป้องกันข้อมูลที่อ่อนไหวด้วย IPSec
- ต้องการมีความปลอดภัยสูงด้วย Smart card
- มีการติดต่อภายนอกพิจารณาเรื่อง Internet authentication

สรุปได้ดังตาราง

ตารางที่ 3.2 แสดง การเปรียบเทียบระดับความปลอดภัย

Basic security	Medium Security	High Security	External Security
Secure e- mail	Secure e- mail	Secure e-mail	Software code signing
EFS	EFS	EFS	Internet authentication
	IPSec	IPSec	
		Smart Card	

3.8.2. การสร้างโครงสร้างพื้นฐาน Certification

หนึ่งในการพิจารณาที่เราต้องดูว่าเมื่อเรานำเข้ามาแล้วจะดำเนินการอย่างไร มีการออกแบบเป็นระดับขั้นหรือไม่ เช่นมี Root CA และมี Intermediate > Issuing ระดับติดต่อผู้ใช้อย่างลงมาเมื่อสร้างโครงสร้างพื้นฐาน CA แล้วเราก็จะตรวจสอบตำแหน่ง และความสัมพันธ์ของ CA ต่างๆการใช้ CAs ภายใน หรือภายนอกอย่างที่เราสามารถที่ทั้ง CA เอง หรือจะใช้ Commercial CA ที่มีอยู่ได้ เราสรุปได้ดังนี้

ตารางที่ 3.3 แสดง การเปรียบเทียบระหว่าง CA ภายในกับ CA ภายนอก

ข้อดีของ CA ภายใน	ข้อเสีย CA ภายใน	ข้อดี CA ภายนอก	ข้อเสีย CA ภายนอก
ควบคุมโดยตรง	เสียแรงในการบริหาร	เพิ่มความมั่นใจกับลูกค้า	ต้นทุนเพิ่ม

	จัดการการรับรอง	ในองค์กร	
ไม่มีค่าใช้จ่ายเพิ่ม	นานวันยิ่งวุ่นวายในการกระจาย	รองรับ สิ่งที่เกิดขึ้นใหม่สำหรับ PKI มีปัญหา	ไม่สามารถลงทะเบียนอัตโนมัติ
สามารถใช้กับ AD	องค์กรต้องรับความรับผิดชอบเมื่อ PKI มีปัญหา	ได้รับคำแนะนำจากช่องที่มีประสบการณ์ และงานด้านกฎหมาย	ไม่ยืดหยุ่นในการจัดการ
อนุญาตให้เพิ่ม PKI ต้นทุนต่ำ	จำกัดกับการติดต่อลูกค้าภายนอก	ลดต้นทุนการบริหาร	จำกัดในการเข้าใช้กับโครงสร้างในองค์กร

ในหลายครั้งเราจะใช้ทั้งสองกรณีคือถ้าเป็นการติดต่อภายในก็จะใช้ CA ภายใน แต่ถ้าต้องการเข้ารหัสภายนอกก็จะใช้ CA ภายนอก

การตั้ง CA สิ่งที่เราพิจารณาหลักคือเราต้องเลือกที่จะลงทะเบียน โดยหนึ่ง CA ของ Windows Server 2003 รองรับการใช้งานได้ถึง 35 ล้านรายการ หรือจะรองรับได้ 2 ล้าน หรือมากกว่าต่อวัน ซึ่งถ้าเรามีหลาย CAs ก็จะเพิ่มจำนวนการร้องขอ และใช้งาน สิ่งที่เราพิจารณามีดังนี้

- จำนวน และความเร็วของ Processors
- ความยาวคีย์
- ประสิทธิภาพดิสก์

ซึ่งจากพื้นฐานต่างๆเหล่านี้ทำให้เราสามารถที่นำไปออกแบบการติดตั้ง CA Server ได้ และถ้าต้องการที่รองรับ Fault tolerance ก็จะต้องมี RAID และถ้าต้องการลดการจราจรใน WAN ก็จะต้องมีการกำหนดการลงทะเบียนที่ท้องถิ่น

3.8.2.1. การสร้าง CA เป็นลำดับขั้น

เราสามารถที่กำหนดความสัมพันธ์ Parent/Child ในการตั้ง CA ใน PKI ได้เป็น Root CA กับ Subordinate CA ใน Subordinate CA ถ้าอยู่ตรงกลางเรียกว่า Intermediate CA และจะติดต่อกับ Root CA อีกทอดหนึ่ง ทั้งหมดเรียกว่า Certificate chain เมื่อเราแสดงค่าที่กำหนดจะมีระดับในแท็บ Certification Path ในองค์กรที่มีการใช้ PKI จำนวนมากก็จะมีโครงสร้างแต่ละระดับมาก หรือมีการกำหนดเป็นลำดับขั้น Root > Intermediate > Issuing CAs เป็นต้น

3.8.2.2. ชนิดของ Windows Server 2003 CA

ในระบบปฏิบัติการ Windows Server 2003 มีอยู่สองแบบคือ

- Enterprise เป็นแบบที่รองรับร่วมกับ Active Directory ซึ่งการทำงานลงทะเลเบียนต้องใช้ควบคู่กับ Active Directory ถูกขายต้องเข้าใช้ Active Directory ได้ ซึ่งไม่เหมาะกับลูกค้านอกองค์กร
- Stand-alone เป็นการที่ไม่ใช้ Active Directory แต่การทำงานทั้งหมดไม่อัตโนมัติต้องกำหนดเองทั้งหมด เหมาะกับการลงทะเลเบียนนอกองค์กรซึ่งถ้าเราวางโครง Enterprise หรือ stand-alone CA เราสามารถที่กำหนดเป็น Root หรือ Subordinate ได้ ซึ่งการกำหนดเหล่านี้ทำให้มีโครงสร้างเป็นลำดับชั้น

3.8.3. การกำหนดค่า Certification

เริ่มจากการออกแบบ การกำหนดค่าที่เป็นโครงสร้าง จะลำดับชั้นหรือไม่ก็ตาม เราต้องมีความเข้าใจในการกำหนดค่าต่างๆดังนี้

- ชนิดของ Certificate เป็นการกำหนดหน้าที่ และการเก็บ Certificate templates ที่ให้กำหนดได้ง่ายกับชนิดที่ระบุในการรับรอง
- การเข้ารหัส และความยาวคีย์ รวมถึงอัลกอริทึม เป็นค่าที่ตรวจสอบ และกำหนดซึ่งมีผลต่อการทำงานของเครื่อง การเลือกระบบการเข้ารหัสหลายวิธีจะทำให้การทำงานช้าลง
- อายุของ Certificate เป็นการกำหนดให้ใช้งานและถ้ากำหนดสั้นไปก็จะต้องกำหนดบ่อย มีผลต่อการจราจร
- นโยบายการต่ออายุ เป็นการกำหนดให้มีการต่ออายุ โดยสามารถที่ใช้คีย์เดิมที่มีอยู่ได้ ซึ่งแนะนำให้ใช้คีย์ใหม่เพื่อความปลอดภัย แต่ก็ยังมีผลต่อประสิทธิภาพ

3.9. การจัดการ Certificates

เมื่อเรามีการออกแบบ โครงสร้าง PKI เรียบร้อยแล้วก็มาสู่ในเรื่องการบริหารงาน ซึ่งมีการลงทะเลเบียน การจัดการ และการประกาศ Certificate revocation list (CRL) เข้าใจเกี่ยวกับการลงทะเลเบียน และการต่ออายุ

CAs ที่มีลูกค้าใช้งานอยู่ จะมีชนิดของ CAs ที่ติดตั้งดังนี้

- ถ้าติดตั้ง enterprise CAs ก็จะสามารถที่ลงทะเลเบียนได้โดยอัตโนมัติ ใน CAs เมื่อได้รับการร้องขอการรับรองจากลูกค้า ก็จะประเมินการให้ใช้งาน หรือการปฏิเสธการใช้งาน

- ถ้าติดตั้ง Stand-alone CAs จะไม่สามารถลงทะเบียนได้อัตโนมัติ ต้องบริหาร หรือเฝ้าดูเอง ใน CA Console สำหรับการรับเข้า และปฏิเสธ

การใช้ลงทะเบียนอัตโนมัติ

เมื่อมีการกำหนดอนุญาตให้มีการร้องขอ และการรับการรับรองจาก CA ซึ่งทำงานอยู่บน Domain controllers ที่ทำงานบน Windows Server 2003 และลูกข่ายเป็น Windows XP Professional โดยมี Enterprise CA ทำงานอยู่ เราสามารถที่ควบคุมการลงทะเบียนได้โดยอัตโนมัติ รวมถึงการกำหนดใช้กับ Group Policy ใน Certificate templates

โดยดีฟอลท์ GPO จะกำหนดค่าลงทะเบียน User และ Computer อัตโนมัติในโดเมน ค่าที่กำหนด Windows Settings\Security Setting\Public Key Policies ทั้งใน User และ Computer configuration

ในกลไกที่ควบคุมการลงทะเบียนเราสามารถที่จัดการได้ใน Certificate Templates ซึ่งจะใช้กำหนดรอบ หรือชนิด ในการลงทะเบียน โดยกำหนดอนุญาตในการร้องขอการรับรองโดยใช้ Templates

เมื่อลูกข่ายเข้ามาร้องขอก็จะตรวจสอบคุณสมบัติใน Active Directory ของออปเจกต์ลูกข่ายว่า กำหนดให้อนุญาตหรือไม่ ถ้าได้รับก็จะลงรับรองให้โดยอัตโนมัติ

การลงทะเบียนด้วยมือ

ในเครื่อง Standalone CAs ไม่สามารถที่รับการลงทะเบียนอัตโนมัติ เราสามารถที่กำหนดการรับเข้ามา และต้องเฝ้าดูเพื่อที่จะให้อนุญาตผู้ใช้ที่ต้องการ ซึ่งค่าที่ร้องขออยู่ในโฟลเดอร์ Pending Requests เมื่อเรออนุญาตก็จะอยู่ในโฟลเดอร์ Issued Certificates

การจัดการร้องขอด้วยมือ

ในกรณีที่มีการร้องขอของ Certificate และรับค่าจากทั้ง Client และ Administrator เราสามารถที่ใช้นั่งในเครื่องมือเหล่านี้ที่ให้มากับ Windows Server 2003

การใช้ Certificates Snap-in

เราสามารถเพื่อดูและจัดการการรับรองทั้งผู้ใช้ และเครื่องที่แสดงในโพลเดอร์ และเราสามารถกำหนดให้ผู้ใช้ หรือเครื่องที่ต้องการได้ ค่าที่กำหนดใช้ Certificate Request Wizard และ Certificate Renewal Wizard

การใช้ Web Enrollment

เมื่อบริการ Certificate ทำงานบนเครื่อง Windows Server 2003 เราสามารถที่ลง IIS และกำหนดให้รองรับกับ ASP เพื่อที่จะสร้างเว็บเพจในการลงทะเบียน CA โดยใช้ผ่านเว็บ ซึ่งอาจจะรองรับการร้องขอจากที่ใดก็ได้ ตำแหน่งที่กำหนดอยู่ที่ http://IP_Address/Certsrv ถ้าเรากำหนดที่ Advanced ก็จะมีค่าชนิด Certificate ที่ให้กรอกในการลงทะเบียน

การยกเลิกการรับรอง

ในกรณีที่ผู้บริหารยกเลิกการรับรอง ค่าคือ Private ยังใช้อยู่ หรือผู้ใช้ที่ไม่มีสิทธิในการใช้ CA หรือต้องการเปลี่ยนค่าพารามิเตอร์ให้แตกต่าง เช่น คีย์ยาวขึ้น เราต้องทำการยกเลิก ซึ่งค่าที่เก็บอยู่ใน CRL

Enterprise CA จะประกาศ CRL ใน Active Directory database ลูกข่ายสามารถใช้การสื่อสารกับ LDAP หรือถ้า Stand-alone CA ที่เก็บ CRL ลูกข่ายต้องใช้ Internet communication protocol เช่น HTTP หรือ FTP ทุกการรับรองจะเก็บ Path ที่กระจายใน CA สำหรับ CRLs เราสามารถที่แก้ไข path ได้ และตรวจสอบจุด CRL ที่ระบุใน Certificate ที่ทำให้มั่นใจว่าการรับรองไม่ได้ถูกยกเลิก ถ้า CRL ไม่มีอยู่ในจุดที่กระจาย แอปพลิเคชันก็จะปฏิเสธการรับรอง

การเลือกโพลเดอร์ Revoked Certificates ใน CA Console และค่าที่แสดงมีการระบุกับ CRL ใหม่ และกำหนด CA ที่ประกาศ delta CRLs ซึ่งในรายการทั้งหมดที่ยกเลิกตั้งแต่ CRL ที่ประกาศสุดท้าย ในองค์กรขนาดใหญ่จะใช้ delta CRLs แทนที่ CRLs ที่เป็นการใช้งานกับแบนด์วิดท์เครือข่าย ซึ่ง delta CRLs ทำทุกสัปดาห์ และประกาศ base CRLs ทุกเดือน

3.10. KeyStore ในภาษาจาวา และ ดอทเน็ต

ในอดีตกการ encrypt และ decrypt message จะใช้ private key และ public key ในการ encrypt และ decrypt message ดังตัวอย่างต่อไปนี้

- Alice กับ Bob ตกลงกันว่าจะใช้ public key algorithm ในการสื่อสาร
- Bob ส่ง public key ของเค้าให้ Alice
- Alice encrypt message ของเธอด้วย Bob public key แล้วส่งไปให้ Bob
- Bob นำ message มา decrypt ด้วย private key ของเขา

จากรูปแบบดังกล่าว public key นี้ยังไม่สามารถเก็บข้อมูลรายละเอียดของเจ้าของ key ได้ว่าเป็นของใคร ใช้ทำอะไร และอื่นๆ จึงมีการออกแบบรูปแบบอิเล็กทรอนิกส์แบบใหม่หรือที่เรียกว่า certificate ซึ่งภายในจะประกอบไปด้วยข้อมูลต่างๆ เช่น รายละเอียดของเจ้าของ key, public key และจุดประสงค์ในการใช้งาน โดยการสร้าง certificate ทำได้โดย

1. สร้าง public key และ private key ของผู้ใช้นั้นมาก่อน (ในการสร้าง key สามารถทำได้โดยหลาย algorithm)
2. สร้าง certificate signing request(CSR) ซึ่งประกอบด้วยรายละเอียดของเจ้าของ key, public key
3. นำ CSR ดังกล่าวส่งไปให้ certificate authority(CA) เป็นคน sign certificate ให้ โดย certificate จะถูกสร้างขึ้นและถูก sign ด้วย private key ของ CA
4. user ได้รับ certificate ดังกล่าวซึ่งพร้อมสามารถใช้งานได้

หลังจากวิธีดังกล่าวข้างต้น เราสามารถรู้ได้ว่า public key ที่เรานำมาใช้เป็นของใครและใช้เพื่ออะไร ตัวอย่างที่เห็นได้ชัดในการใช้ certificate เช่น SSL (secure socket layer) สำหรับการสร้าง certificate นั้นเราอาจจะไม่จำเป็นต้องให้ CA เป็นผู้ sign certificate ให้กับเราก็ได้ โดยเราสามารถที่จะใช้ private key ของเราเป็นตัว sign certificate ขึ้นมาเองได้ โดยวิธีนี้เรียกว่า self-sign

โดยปกติการสร้าง key entries ของภาษายาวานั้น key entries ที่ได้จะถูกเก็บอยู่ใน KeyStore โดย KeyStore นี้เป็น interface ของภาษายาวา ส่วน KeyStore implementations จะขึ้นอยู่กับ provider แต่ละ security provider เป็นผู้กำหนดว่าจะมี KeyStore ประเภทไหนบ้าง โดยปกติแล้ว sun provider ได้มีการ implement KeyStore ซึ่งเป็น Type JKS มากับ j2se (java 2 standard edition) สำหรับ provider package อื่นๆ ได้มีการ implement KeyStore ในรูปแบบอื่นๆไว้ โดยรูปแบบที่เป็นมาตรฐานรูปแบบหนึ่ง คือ PKCS12 ซึ่งสามารถหาใช้งานได้จาก provider ตัวอย่างเช่น provider package Bouncy Castle (www.bouncycastle.org) ซึ่งเป็น opensource โดยมี KeyStore เพิ่มเติมให้ใช้เช่น BKS, UBER, PKCS12 โครงสร้างของ key entries จะประกอบด้วย public key, private key และ certificate ถูกใส่ (chain

certificate) นอกจากนี้ KeyStore จะเก็บ trusted certificate entries ซึ่งภายในจะเก็บ single public key certificate ของ CA สำหรับรูปแบบของ keystore แบบต่างๆ มีดังต่อไปนี้

- JKS เป็นรูปแบบ keystore ที่สามารถเก็บ private key และ certificate ซึ่งมี public key บรรจุอยู่ได้หลายชุด โดยใช้วิธีการเก็บแบบ map โดยมี alias เป็นตัวระบุชื่อ(ก็ยในการ map) ส่วน component อื่นๆจะเป็น value สำหรับ alias นั้นๆ โดยแต่ละ component จะมี password ของคีย์ซึ่งสามารถกำหนดได้ และจะต้องมี password ในระดับ KeyStore ด้วย
- PKCS12 (personal information exchange syntax standard) รูปแบบนี้เป็นรูปแบบที่นิยมใช้ทั่วไป ตัวอย่างแอปพลิเคชันที่รองรับรูปแบบนี้ เช่น certificate manager ของ windows, OpenSSL รูปแบบจะเก็บ 1 key component ต่อ 1 KeyStore เท่านั้น และแต่ละ KeyStore จะต้องมีการมี password เพื่อป้องกันการเข้าถึง private key
- BKS (Bouncy Castles's version ของ JKS) เป็น KeyStore ที่มีรูปแบบโครงสร้างเหมือนกับ JKS สามารถทำงานร่วมกับ keytool ได้ keystore ชนิดนี้จะป้องกันการแก้ไข แต่ไม่ป้องกันการตรวจสอบ
- UBER (Bouncy Castles's secure version ของ BKS) เป็น KeyStore ที่สามารถทำงานร่วมกันกับ keytool ได้เท่านั้นถ้า password ถูกกำหนดจาก command line เนื่องจาก key ทั้งหมดถูก encrypt โดย PBKWithSHAAndTwofish-CBC วิธีนี้จะทำให้ KeyStore สามารถสกัดกั้นการแก้ไขและตรวจสอบได้ และมีการบังคับให้มีการตรวจสอบเสมอ
- JCEKS เป็น KeyStore ของ SUN JCE provider เริ่มมีใน j2sdk version 1.4 เพื่อเพิ่มความสามารถในการป้องกัน private key ได้ดีกว่า KeyStore JKS ซึ่งเป็นของ SUN provider โดยใช้ password-based encryption with Triple DES ในการป้องกันแต่ละ key entries แต่ JKS ใช้เพียงแค่ hash value ในการป้องกันแต่ละ key entries ข้อแตกต่างจาก JKS อีกข้อหนึ่งคือ JCEKS มันจะเก็บ private key หรือ secret key อย่างใดอย่างหนึ่งไว้ใน key entries
- ถ้ามีการใช้ Java Cryptography Extensions(JCE) จะไม่มีการใช้ JKS ส่วน JCEKS จะใช้เมื่อมีการใช้ JCE ถึงแม้ว่า JCEKS จะมีความปลอดภัยสูงกว่าแต่ประสิทธิภาพของมันก็น้อยลงไปด้วย

รูปแบบของ key มีดังนี้ JKS ,JCEK,PKCS12,JCERACFKS(z/OS) ซึ่งรูปแบบ JKS จะไม่เก็บ shared key ส่วน JCEK เป็นรูปแบบที่มีความปลอดภัยสูง และรูปแบบ PKCS12 เป็นรูปแบบของ key มาตรฐาน

KeyStore ที่กล่าวมาข้างต้นที่นอกเหนือจาก JKS สามารถใช้กับ utility keytool ได้ทั้งหมด โดยจะต้องกำหนดพารามิเตอร์ `-storetype`, `-providerName`, `-providerClass` และ `-providerArg` ให้เหมาะสมกับแต่ละ keystore โดยสามารถหาข้อมูลเพิ่มเติมได้จาก security provider document สำหรับ keystore ที่จะใช้

การสร้าง KeyStore นั้น สามารถทำได้สองวิธี คือ การเขียนโปรแกรมโดยใช้ `java.security` package หรือใช้ keytool ซึ่งเป็น management utility ในการ generate public key และ private key ขึ้นมานั้นจะต้องมี key algorithm และ signature algorithm ซึ่งสองอย่างนี้จะขึ้นอยู่กับแต่ละ provider ว่ามี algorithm แบบไหนให้ใช้บ้าง โดยปกติแล้ว key algorithm ที่ใช้มี 2 ชนิด คือ RSA กับ DSA ที่นิยมใช้ ส่วน signature algorithm ถ้า key algorithm เป็น DSA มักจะใช้ signature algorithm เป็น SHA1withDSA ส่วน RSA มักจะใช้ signature algorithm เป็น MD5withRSA

สำหรับการสร้าง KeyStore แบบ JKS โดยใช้ keytool ทำได้ดังต่อไปนี้

```
keytool -genkey [-alias <alias>] [-keysize <keysize>] [-keyalg <keyalg>] [-keystore keystore]
```

<alias> คือ ชื่อของแต่ละ key component โดย ค่า default จะเป็น mykey

<keysize> คือ ขนาดของ key ที่ gen ขึ้นมา โดยถ้าเป็น DSA ใช้ได้แค่ 512 ถึง 1024 โดยต้องเป็นค่าที่หารด้วย 64 งตัว สำหรับ RSA สามารถตั้งแต่ 512 ถึง 4098

<keyalg> เป็นได้ทั้ง RSA และ DSA แต่ค่า default เป็น DSA

<keystore> เป็นการระบุ file KeyStore ที่สร้างขึ้นว่าอยู่ไหน โดยปกติแล้ว file ที่ถูกสร้างขึ้นจะถูกเก็บไว้ใน user directory ของแต่ละ operating system โดยมีชื่อ file ว่า .keystore ตัวอย่างเช่น

```
keytool -genkey -alias mykey -keyalg RSA -keysize 1024 -keystore keystore.jks
```

จากตัวอย่างเป็นการสร้าง key โดยให้ key มี alias เป็น mykey , key algorithm ที่ใช้เป็น RSA, key มีขนาด 1024 บิต และเก็บ keystore ที่ได้ไว้ในไฟล์ keystore.jks

สำหรับการสร้าง KeyStore แบบอื่นๆ สามารถใช้ keytool ได้ แต่ต้องมี provider class ของ key store แบบอื่นๆ ไปด้วย เช่น PKCS12 ซึ่งการสร้างแบบนี้ไม่นิยมใช้ keytool สร้าง แต่นิยมใช้การเขียน code gen ขึ้นมาเลย

บทที่ 4

การทดลองและผลการทดลอง

4.1. โครงร่างของระบบ

4.1.1. ฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในการทดลอง

เครื่องคอมพิวเตอร์ที่ใช้ในการทดลอง

ASP .Net เว็บแอปพลิเคชันเซิร์ฟเวอร์

หน่วยประมวลผลกลาง	: AMD Sempron 2000+
หน่วยความจำ	: 512 MB
ฮาร์ดดิสก์	: 80 GB
ระบบปฏิบัติการ	: Windows OS
ระบบฐานข้อมูล	: PostgreSQL version 8.0
เซิร์ฟเวอร์	: IIS version 5.1
ทูลที่ใช้พัฒนา	: Microsoft Visual Studio .Net 2003
Plug-in	: WSE 1.0

Java Web Service Server

หน่วยประมวลผลกลาง	: Pentium4 1.7 GHz
หน่วยความจำ	: 512 MB
ฮาร์ดดิสก์	: 120 GB
ระบบปฏิบัติการ	: Windows OS
เซิร์ฟเวอร์	: Websphere Application Server 5.1
ทูลที่ใช้พัฒนา	: Websphere Application Developer 5.1.1

Database Server

หน่วยประมวลผลกลาง	: Pentium M 1.4
หน่วยความจำ	: 512 MB
ฮาร์ดดิสก์	: 40 GB
ระบบปฏิบัติการ	: Windows OS
ระบบฐานข้อมูล	: MySQL version 3.2 build 6.14

ทูลที่ใช้พัฒนา : Eclipse 3.1

.Net Web Service Server

หน่วยประมวลผลกลาง : Pentium 4 3.0 GHz

หน่วยความจำ : 1 GB

ฮาร์ดดิสก์ : 120 GB

ระบบปฏิบัติการ : Windows OS

ระบบฐานข้อมูล : PostgreSQL version 8.0

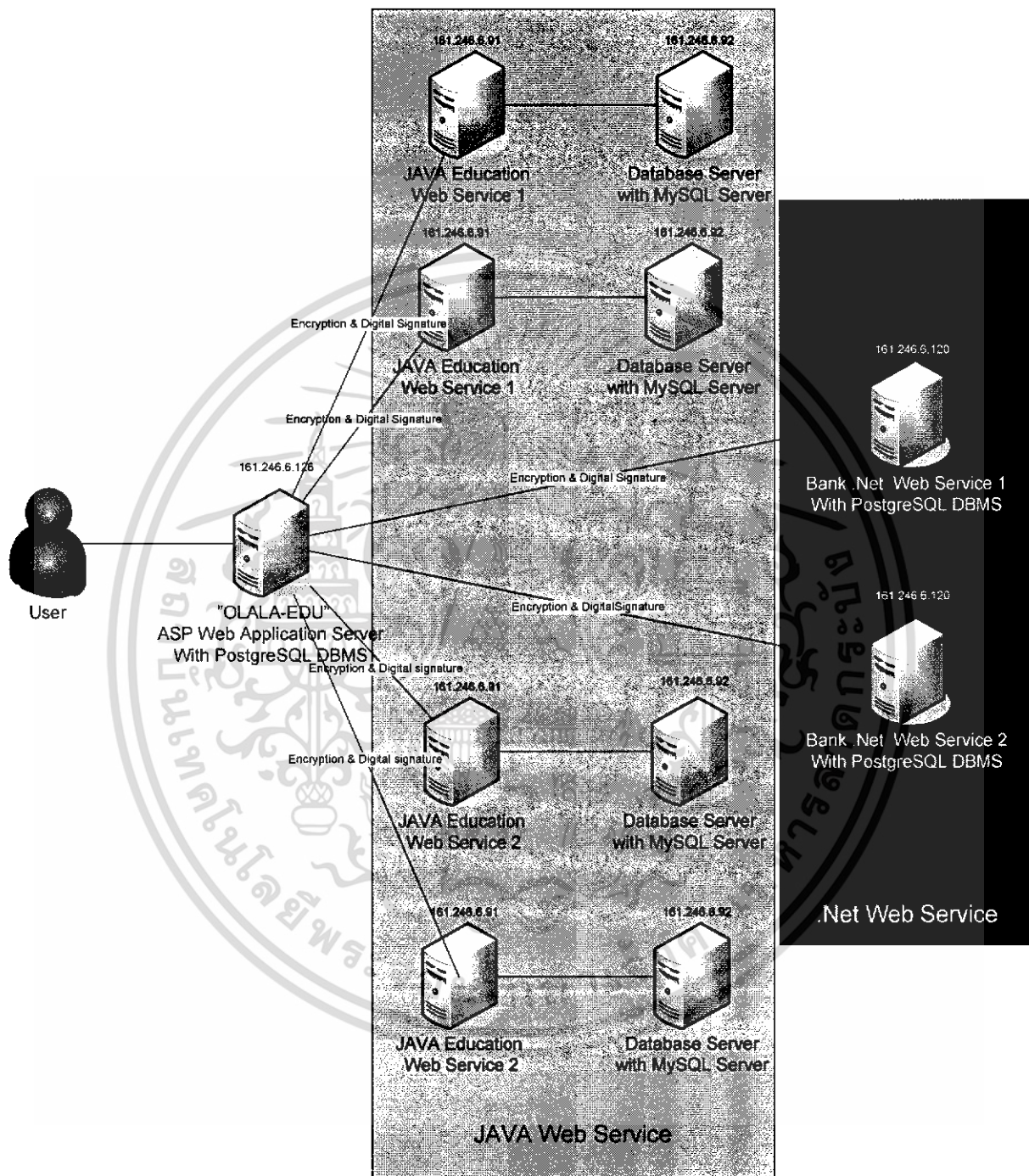
เซิร์ฟเวอร์ : IIS version 5.1

ทูลที่ใช้พัฒนา : Microsoft Visual Studio .Net 2003

Plug-in : WSE 2.0



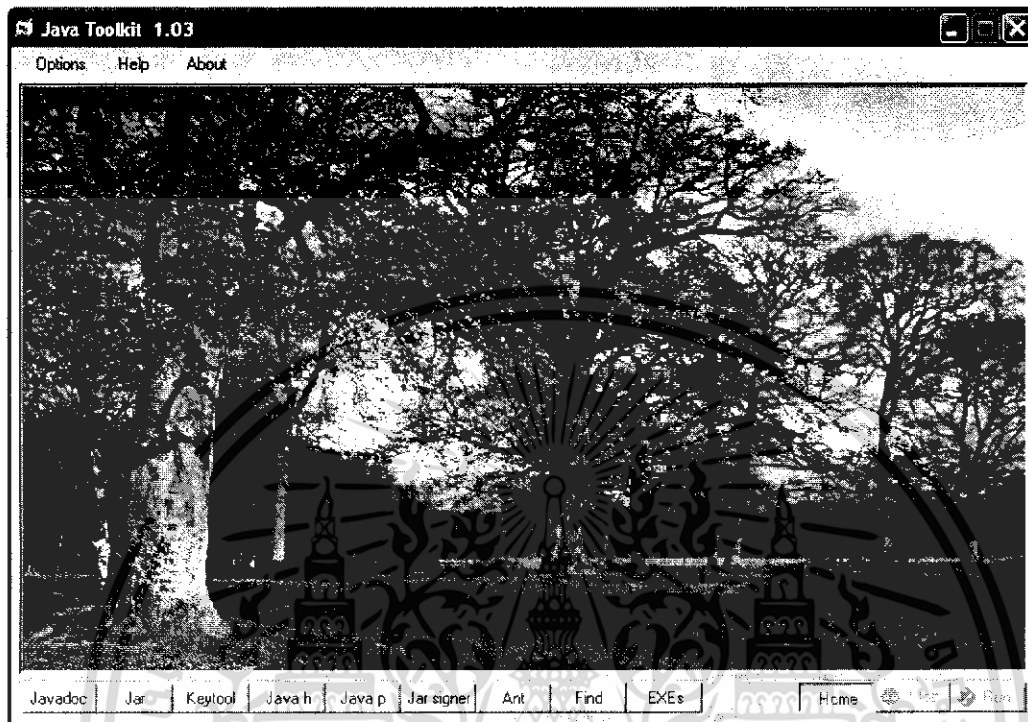
4.1.2. แบบจำลองระบบที่มีการนำ WS-Security ไปใช้



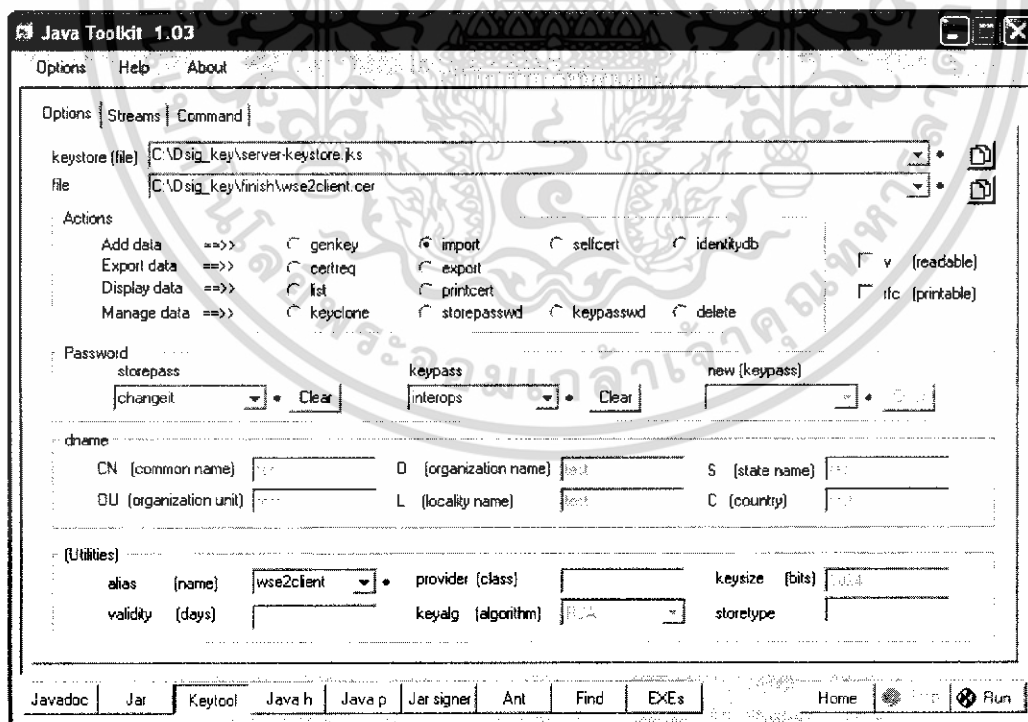
รูปที่ 4.1 แสดงแบบจำลองระบบที่มีการนำ WS-Security ไปใช้

4.2. ผลการทดลอง

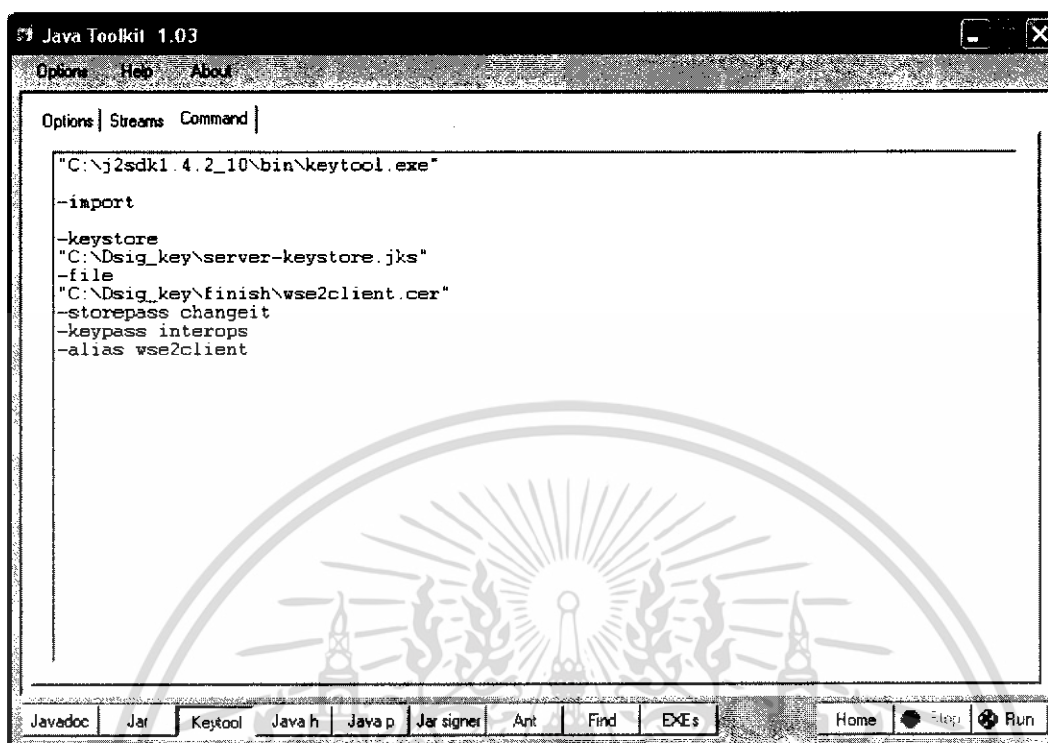
4.2.1. การสร้างคีย์สำหรับใช้ในเว็บเซอร์วิสด้วย Java Toolkit 1.03



รูปที่ 4.2 แสดง โปรแกรม Java Toolkit



รูปที่ 4.3 แสดงการสร้างคีย์ด้วยโปรแกรม Java Toolkit



รูปที่ 4.4 แสดงผลการสร้างคีย์ด้วยโปรแกรม Java Toolkit เป็น command

คีย์ที่ใช้ในการจัดการเรื่องความปลอดภัยของเว็บเซอร์วิสมีหลายรูปแบบซึ่งคีย์ที่ใช้ร่วมกันได้มีเพียงคีย์รูปแบบ PKCS12 เท่านั้นที่ใช้ได้ทั้งคู่ ซึ่งสามารถอธิบายรูปแบบการสร้าง อย่างง่าย ได้ ดังนี้

ตารางที่ 4.1 แสดงรูปแบบการสร้าง คีย์แบบต่างๆ

ชนิดของคีย์	รูปแบบการสร้าง	การนำไปใช้
JKS	1. สร้างเองจาก keytool -genkey 2. import จากคีย์ชนิดอื่น โดยใช้ keytool -import	Java Platform เท่านั้น
JCEKS	1. สร้างเองจาก keytool -genkey 2. import จากคีย์ชนิดอื่น โดยใช้ keytool -import	Java Platform เท่านั้น
PKCS12	สร้างจาก Tool เฉพาะซึ่งมีความน่าเชื่อถือ เช่น OpenSSL	Java & .Net สามารถใช้ได้
PFX	1. สร้างเองจาก makecert tool 2. ทำการconvert key ของ Java โดยใช้ OpenSSL	.Net เท่านั้น
CER (public key)	1. สร้างได้จาก makecert tool 2. สามารถขอคีย์จาก CA ได้	Java & .Net สามารถใช้ได้

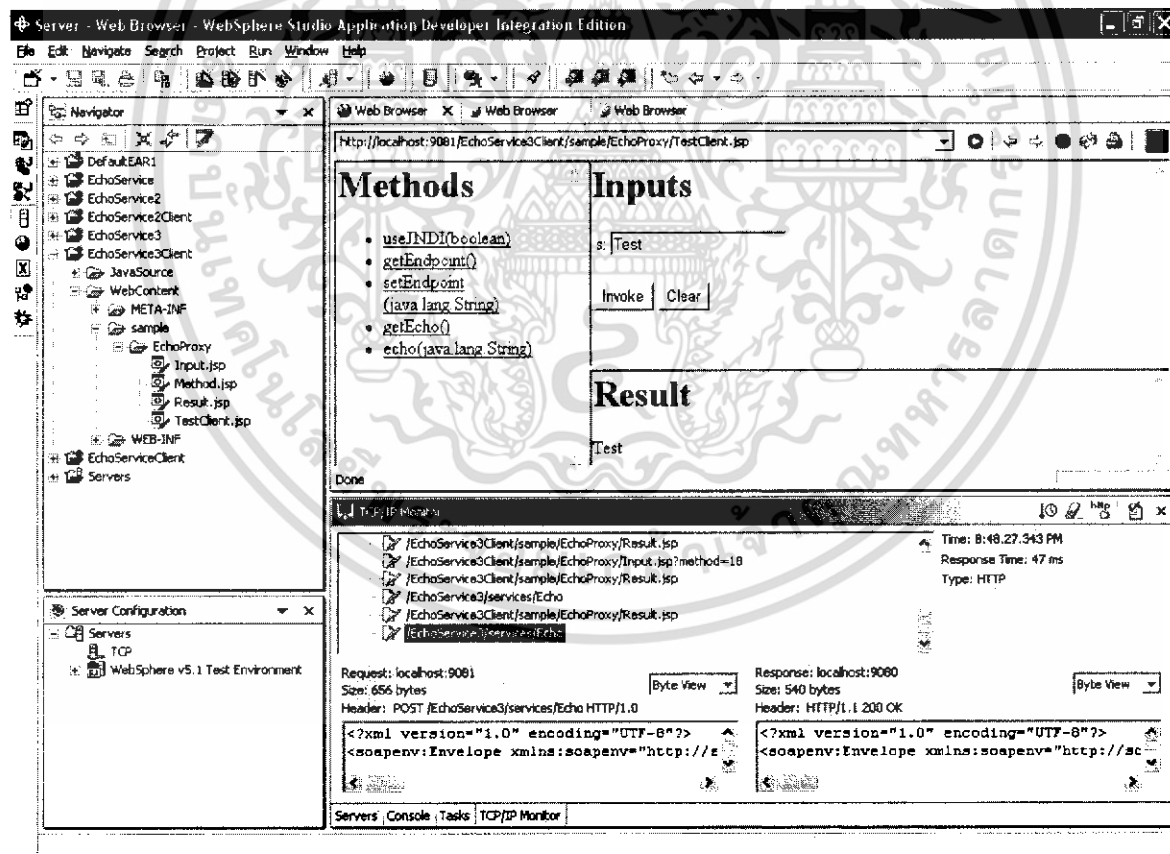
1. คีย์ที่จะนำไปใช้ใน Java web service ได้นั้นต้องถูกบรรจุใน “Keystore” ชนิด JKS, JCEKS, PKCS12 เท่านั้นโดยสามารถสร้างขึ้นเองโดยใช้ keytool -genkey หรือ นำคีย์ที่มีอยู่แล้ว import เข้าไปใน keystore ก็ได้

2. คีย์ที่จะนำไปใช้ใน .Net Webservice ต้องเป็นคีย์ที่ถูกบรรจุใน “Keystore” ชนิด PKCS12, PFX หรือ public key certificate โดยสามารถสร้างจาก makecert หรือ อาจนำคีย์ของจาวา มาใช้โดยใช้ OpenSSL ในการแปลงคีย์

4.2.2. การเซตค่าเพื่อให้สอดคล้องกับหลักการ WS-Security ระหว่าง Java Web service และ Java Webservice โดยใช้ Websphere Application Developer 5.1.1

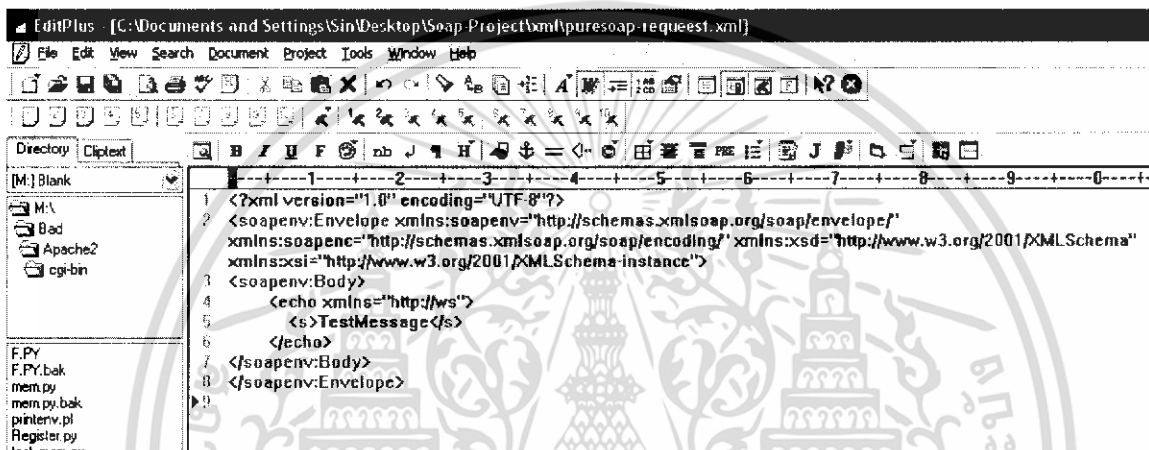
จากทฤษฎีเกี่ยวกับเว็บเซอร์วิสที่ผ่านมาได้ทำการทดลองใช้งานในส่วนของ WS-Security ซึ่งแสดงให้เห็นใน SOAP ได้ดังตัวอย่าง

4.2.2.1. การเพิ่มความปลอดภัยในการพิสูจน์ตัวตน (Authentication) โดยใช้ Username Token

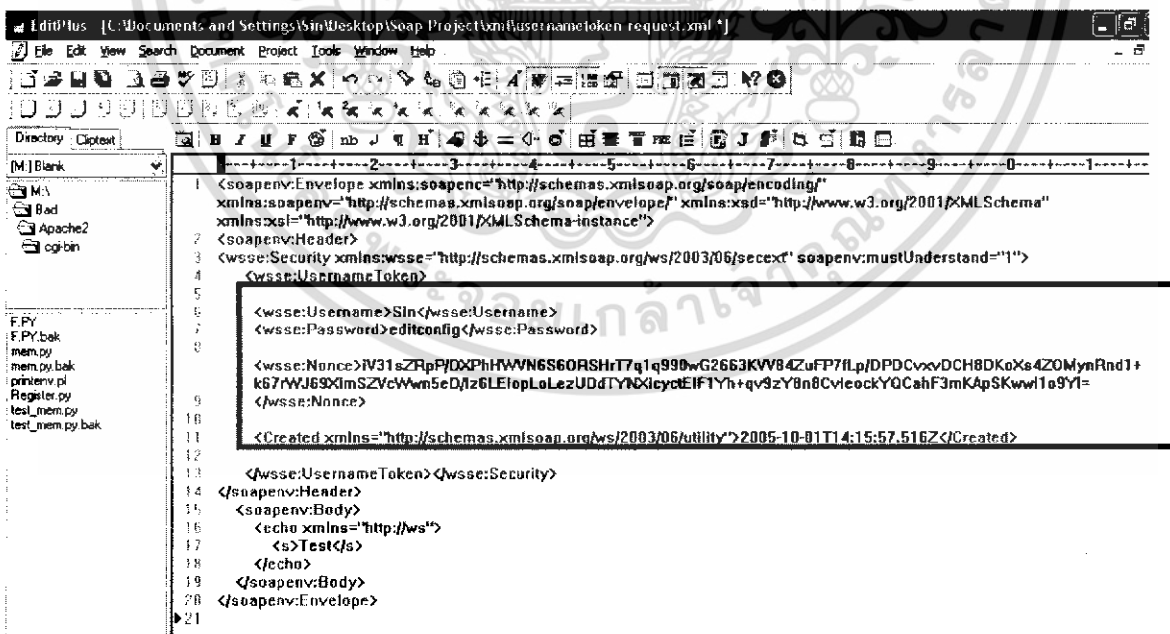


รูปที่ 4.5 แสดง ตัวอย่างโปรแกรม Echo Service ที่มีการทำ Username Token

ใน SOAP ที่มีการทำ Username Token แล้วจะมี Tag <wsse: usernameToken> แสดงถึงการใส่ Username Token ใน SOAP นั้น เพื่อหลีกเลี่ยงการโจมตี จึงมีการสร้าง element <wsse:Nonce> และ <wsu:Created> ไว้ด้านใน <wsse: usernameToken> element และใช้ในการรับรอง message ในการร้องขอผู้รับหรือการตอบกลับผู้รับ จะทำการตรวจสอบ ความบริสุทธิ์ ว่ามีความแตกต่างจาก message ที่ถูกสร้างไว้ก่อนหน้านี้หรือไม่

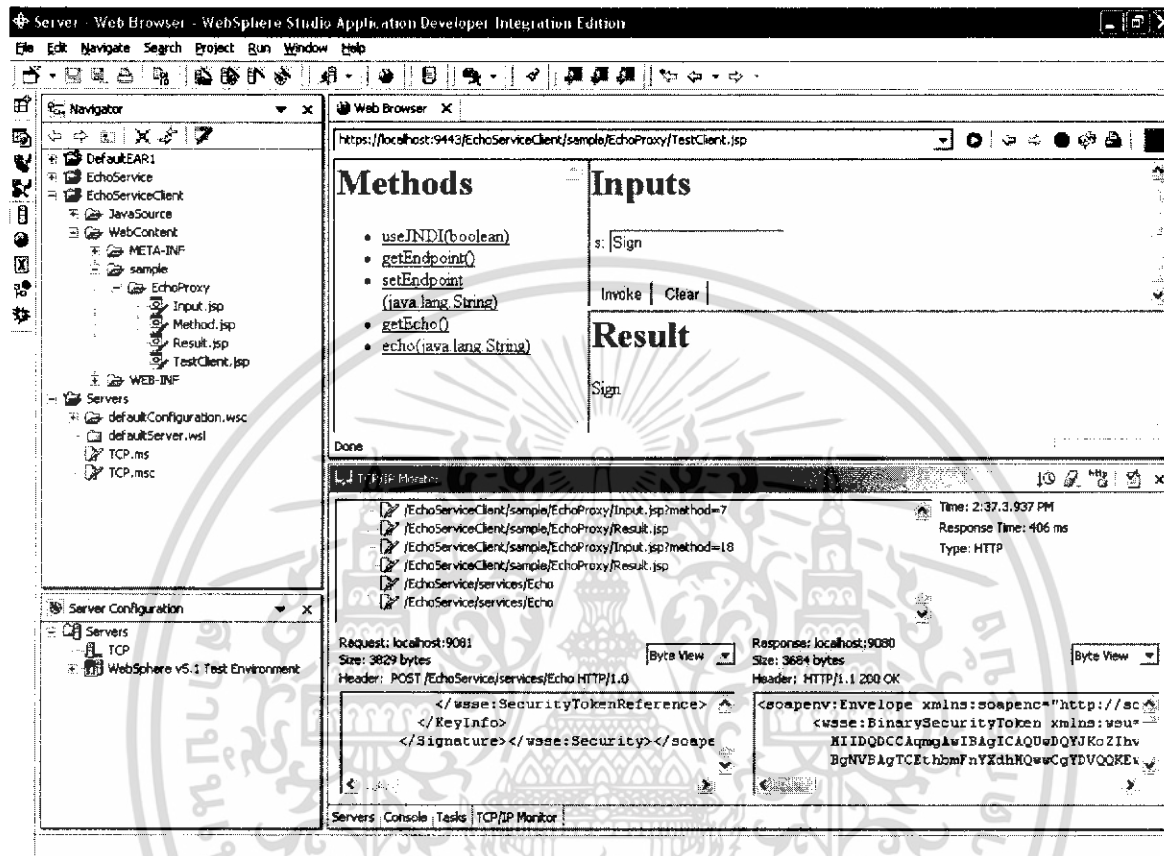


รูปที่ 4.6 แสดง SOAP ที่ยังไม่ได้มีการทำ Username Token

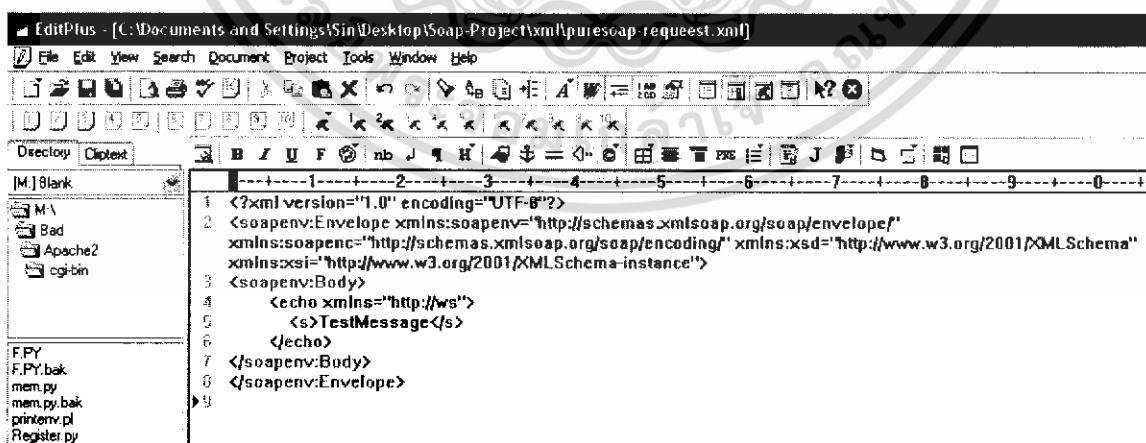


รูปที่ 4.7 แสดง SOAP ที่มีการทำ Username Token แล้ว

4.2.2.2. การรักษาความสมบูรณ์ของข้อมูล (Integrity) โดยการใช้ Digital Signature



รูปที่ 4.8 แสดงตัวอย่างโปรแกรม Echo ที่มีการทำ Digital Signature แล้ว



รูปที่ 4.9 แสดง SOAP Message ที่ไม่มีการ ทำ Digital signature

soap-sign-request.xml

10/3/2005 11:25 AM

```

1 <soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
2 <soapenv:Header>
3 <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext"
  soapenv:actor="myActorURL" soapenv:mustUnderstand="1" >
4 <wsse:BinarySecurityToken xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
  EncodingType="wsse:Base64Binary" ValueType="wsse:X509v3"
  wsu:Id="wssecurity_binary_security_token_id_408039426707555435" >
5 MLIIDQTCAGggAwIBAgICAgQwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCSEAETAP
6 BgNVBAGTCCEhbmF0YXZlbnQwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCSEAETAP
7 AUMHSW50IENBMjAcFw0wMTEwMDUwOTU0MDZaFw0wMTEwMDUwOTU0MDZaMFQxOzA1BgNV
8 BAYTAkpQMRQwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCSEAETAP
9 UKwxFjAUBgNVBAITDVBhbmF0YXZlbnQwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCSEAETAP
10 </wsse:BinarySecurityToken>
11 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#" >
12 <SignedInfo>
13 <CanonicalizationMethod
14 Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" ></CanonicalizationMethod>
15 <SignatureMethod
16 Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" ></SignatureMethod>
17 <Reference URI="#wssecurity_body_id_4508207003845519626" >
18 <Transforms>
19 <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" >
20 <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
21 PrefixList="xsi soapenc xsd #default wsu soapenv" ></ec:InclusiveNamespaces>
22 </Transform>
23 </Transforms>
24 <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" ></DigestMethod>
25 <DigestValue>ml0014nv909fA61S687s0n0rD0= </DigestValue>
26 </Reference>
27 </SignedInfo>
28 <SignatureValue>
29 g1Ez12JQMDc14OLktdLLpfAw7Xr3s9GoXaLnT7zenPRjHhMgmNnLH5
30 a/VVOSqTYFhX6vJmukPFu83jTVdzPeT8/BoOFwlu/War6vdKSSuXcPd9
31 Vc0RC+dkGV2Fcs33+EBvXaXKvSS6147pGUMENaueM0DzWt8=
32 </SignatureValue>
33 <KeyInfo>
34 <wsse:SecurityTokenReference>
35 <wsse:Reference
36 URI="#wssecurity_binary_security_token_id_408039426707555435" ></wsse:Reference>
37 </wsse:SecurityTokenReference>
38 </KeyInfo>
39 </Signature>
40 </wsse:Security>
41 </soapenv:Header>
42 <soapenv:Body xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility"
  wsu:Id="wssecurity_body_id_4508207003845519626" ><echo xmlns="http://ws"><s>Test
  Message</s></echo>
43 </soapenv:Body>
44 </soapenv:Envelope>

```

รูปที่ 4.10 แสดง SOAP Message ที่มีการ ใช้ Digital signature

จากภาพแสดงให้เห็นส่วนประกอบที่เพิ่มขึ้นมาหลังจากมีการ ทำ Digital signature แล้ว คือใน ส่วนของ element ต่างๆ

< Signature >

เป็นส่วนที่เก็บข้อมูลทั้งหมดโดยกล่าวถึง Information ที่ใช้ในการ sign เช่น signature, key ที่ ใช้ sign, ตำแหน่งที่ใช้ในการเก็บ information

< SignedInfo >, < SignatureMethod > , < CanonicalizationMethod >

เป็นส่วนที่ระบุชนิดของ Signature เช่น Kerberos หรือ RSA ซึ่งใช้ในการสร้าง signature element เหล่านี้ เป็นส่วนที่แสดงให้เห็นว่ามีการแบ่งการทำงานเป็นส่วนๆ และสามารถแสดงให้เห็นถึงการป้องกันการเปลี่ยนแปลงแก้ไขข้อมูล จากบุคคลที่ 3

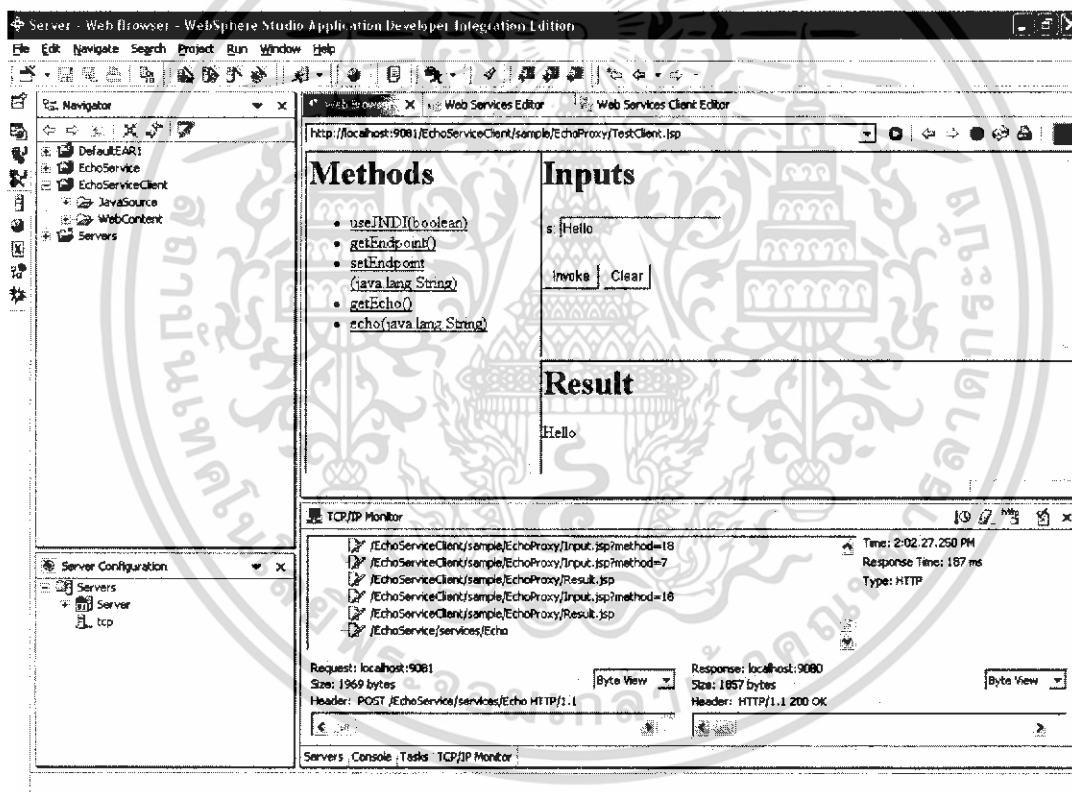
< SignatureValue >

เป็นส่วนที่เก็บข้อมูลรายละเอียดของ Application

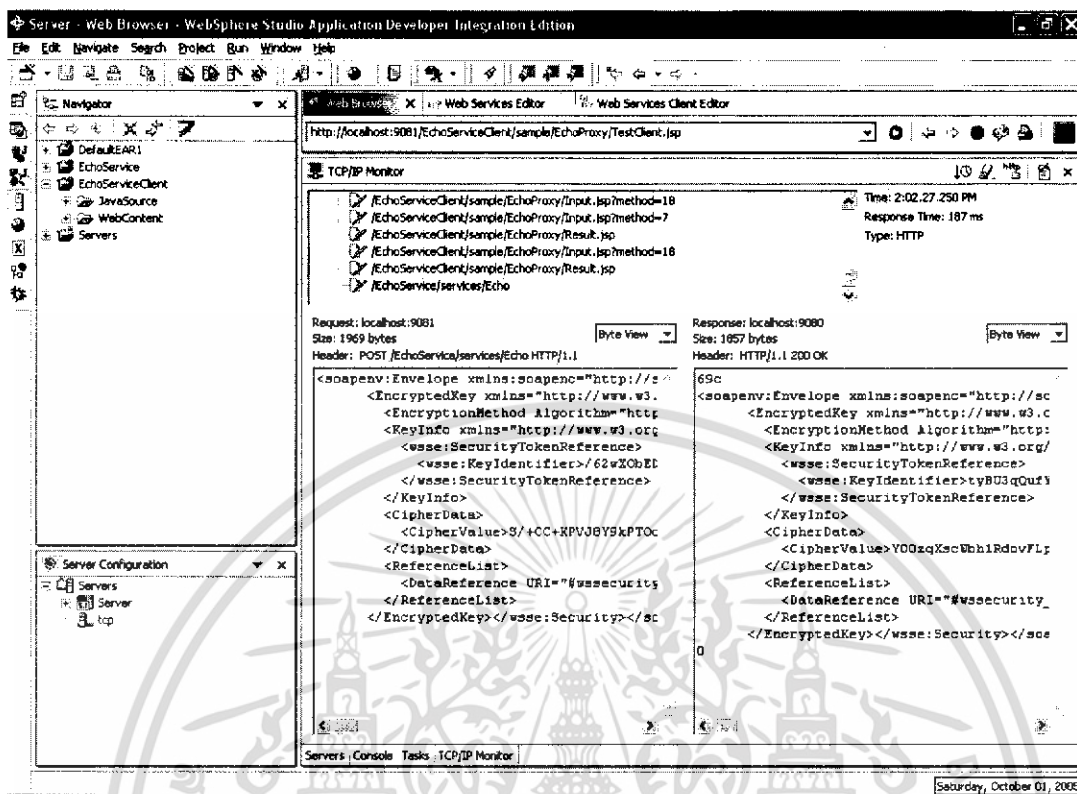
< KeyInfo >

เป็นส่วนที่ระบุชื่อ หรือ Key ที่ใช้ sign

4.2.2.3. การเพิ่มความปลอดภัยของข้อมูลที่เป็นความลับ (Confidentiality) โดยการใช้ XML Encryption 3 รูปแบบ



รูปที่ 4.11 แสดง การ ส่ง Echo Message ที่มีการ Encryption SOAP เข้าไปด้วย



รูปที่ 4.12 แสดง การ ส่ง Echo Message ที่มีการ Encryption SOAP เข้าไปด้วย

Order.xml 10/3/2005 1:44 PM

```

1 <?xml version="1.0"?>
2 <!-- This Listing provides the sample XML File that will be encrypted. -->
3
4 <purchaseOrder>
5   <Order>
6     <BookName>
7       SoccerForDummies
8     </BookName>
9     <Id>123-958-74598</Id>
10    <Quantity>500</Quantity>
11  </Order>
12
13  <Payment>
14    <CardNo>4502-3456-3278-2011</CardNo>
15    <CardType>VISA</CardType>
16    <ValidDate>12-10-2004</ValidDate>
17  </Payment>
18
19 </purchaseOrder>

```

รูปที่ 4.13 แสดง ตัวอย่าง XML ที่มีหลาย element

4.2.2.3.1. การเข้ารหัสทั้งเอกสาร

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#" Id="Test"
  Type="http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml">
4   <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
5   <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
6     <KeyName>theKey</KeyName>
7   </ds:KeyInfo>
8   <CipherData>
9     <CipherValue>v0oy2cty7WLj5RDNbV2o3UG/BtTt08IWWXf0Y/EHu/qivSa/p7wXP03mSXvtaGY87 +
    hvjCrrH4mp
10    cKr+0xrTJumhYKHyptrHTpJksHxehqM9tCZyQDR0rZeLo76KrU2/FDS22DS3SLYDda3PUf5v0paU
11    k2TdoCJIttXPwvbvULpzvP8/5BYVT+t8bqm24YEC9Q8shlmAc/OdXV03zfWIMKoqAZj1je03aImR
12    BobTnqd1o+ilimFUEV6Dmg0pdwDuu+CsN9J3wE3C4co6ISh6zEqlFWmJA2wnytGTi6/t24fzRHeK
13    7f+SOIC+7yLjnnHPBfpvp2ciC/IEg5jHd2qg4vhSKN0k6qfJijpc5M4lk4dE79lv7LpF5m22tf8
14    z1pDXmwigkiYVfi3ksIPXmHALii/y4ggx4u2pCAFR17Bk4bxRL4INpoJx/ykl8JfidBrGvKiuYB
15    nbdnCn/BjKzRUyyTfcgHQXfCeUe7JrueiQJjiTIPwI3Ye8u4fipDfegCnbV/rb5/a7vFe2Z/RfRq
16    xUPBJ09K0uP4+DMTP3qAfkl8F0qqQ5UnvnbLFYNxnx4D</CipherValue>
17   </CipherData>
18 </EncryptedData>
19
20

```

รูปที่ 4.14 แสดง SOAP Message มีการเข้ารหัสทั้งเอกสารแบบ triple-DES

4.2.2.3.2. การเข้ารหัสเฉพาะ Element

```

4.xml 10/3/2005 1:53 PM
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!-- This Listing provides the sample XML File that will be encrypted. -->
4 <purchaseOrder>
5   <Order>
6     <BookName>Soccer For Dummies</BookName>
7     <Id>123-958-74598</Id>
8     <Quantity>500</Quantity>
9   </Order>
10  <Payment>
11    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#" Id="Test"
      Type="http://www.w3.org/2001/04/xmlenc#Content">
12      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
13      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
14        <KeyName>theKey</KeyName>
15      </ds:KeyInfo>
16      <CipherData>
17        <CipherValue>6n8juQbwTMrXB/3RNvvyul+tCLP1YxYf4m9XUNDwQT+fj7TfV2yW45YvALTyCtHU
        mmu8zj200iPb
18        47F/18f8lnhkvo0aVSLpKJsNx10oWW23sKnLyvb6sou2Fg0Gx0NgMUbwgui3ALM=</CipherValue>
19      </CipherData>
20    </EncryptedData>
21  </Payment>
22 </purchaseOrder>
23
24

```

รูปที่ 4.15 แสดงการเข้ารหัส เฉพาะ Element < Payment > แบบ tripledes

4.2.2.3.3. การเข้ารหัสเฉพาะข้อมูลในเอกสาร

3.xml

10/3/2005 2:00 PM

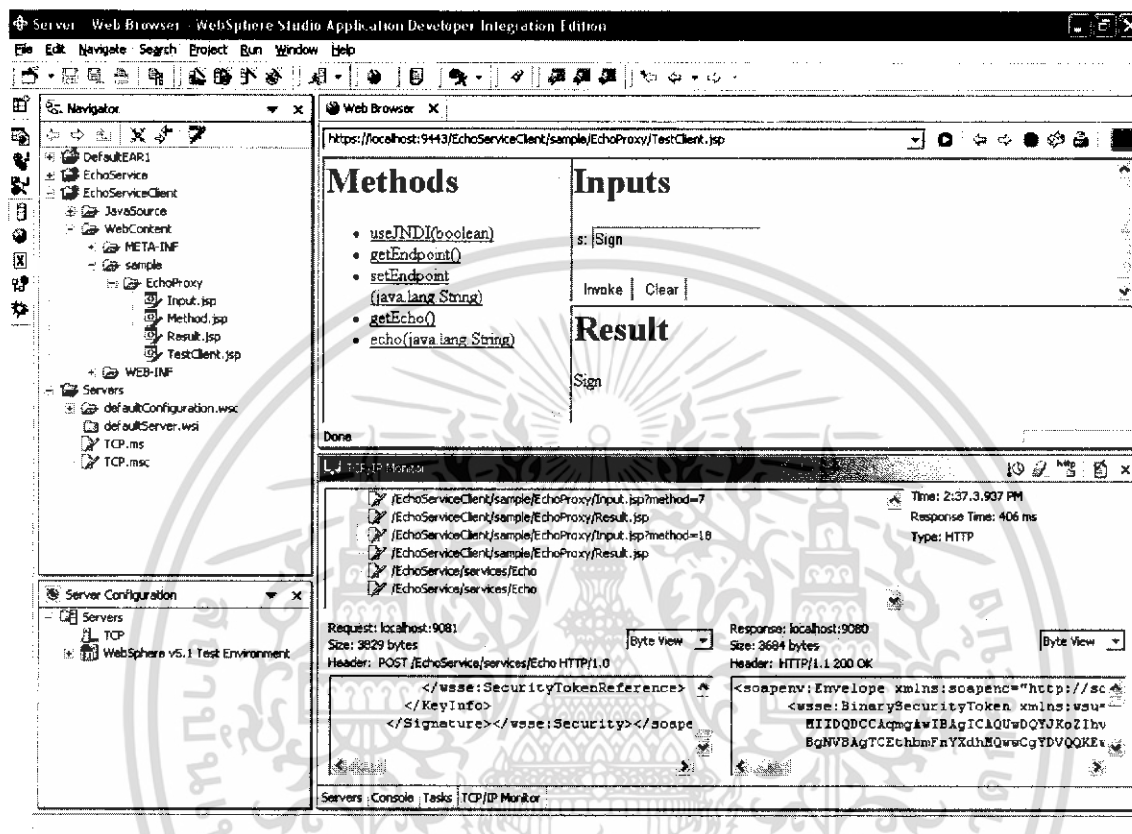
```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!-- This Listing provides the sample XML File that will be encrypted. -->
4  <purchaseOrder>
5    <Order>
6      <BookName>Soccer For Dummies</BookName>
7      <Id>123-958-74598</Id>
8      <Quantity>500</Quantity>
9    </Order>
10   <Payment>
11     <CardNo>
12       <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#" Id="Test"
13       Type="http://www.w3.org/2001/04/xmlenc#Content">
14         <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
15         <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
16           <KeyName>theKey</KeyName>
17         </ds:KeyInfo>
18         <CipherData>
19           <CipherValue>d05qh+vPyydTPfmrAuhkP208YeuD+N+JT56ALkxj9eY=</CipherValue>
20         </CipherData>
21       </EncryptedData>
22     </CardNo>
23     <CardType>VISA</CardType>
24     <ValidDate>12-10-2004</ValidDate>
25   </Payment>
26 </purchaseOrder>
27

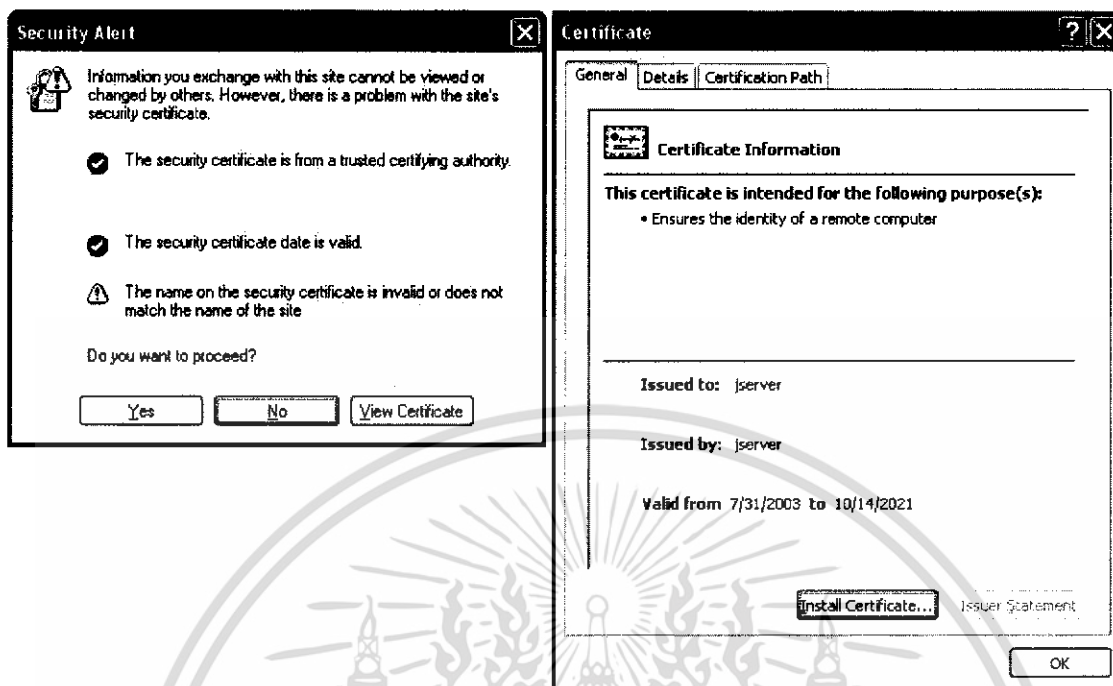
```

รูปที่ 4.16 แสดงการเข้ารหัสเฉพาะข้อมูลใน < CardNo > แบบ tripledes

4.2.2.4. การเพิ่มความปลอดภัยในการขนส่งข้อมูล (Transport – level Security) โดยการนำ SSL ไปใช้ในเว็บเซิร์ฟเวอร์



รูปที่ 4.17 แสดง Echo Service ที่มีการ Configure SSL ที่ พอร์ต 9443



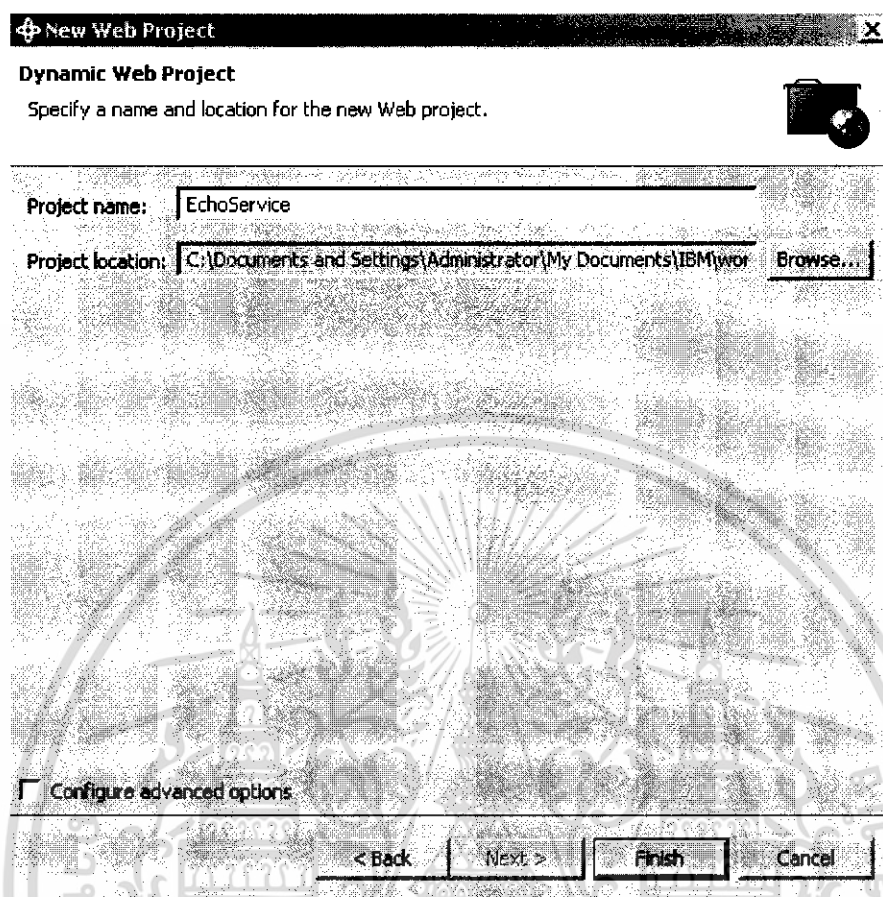
รูปที่ 4.18 แสดง Certificate และ แสดงว่า มีการติดตั้ง SSL บน Server เรียบร้อยแล้ว

4.2.3. การเซตค่าเพื่อใช้ในการทำ WS-Security แบบ Encryption ระหว่าง Websphere Application Developer 5.1 และ Microsoft Visual Studio .Net 2003

4.2.3.1 การเซตค่าของ Websphere Application Developer 5.1

4.2.3.1.1 การสร้างเว็บเซอร์วิส

1. สร้างเว็บเซอร์วิสโดยเลือกที่ File->New->Web->Dynamic Web Project จากนั้นทำการตั้งชื่อโปรเจค ดังตัวอย่าง ในที่นี้ใช้ Echo Service

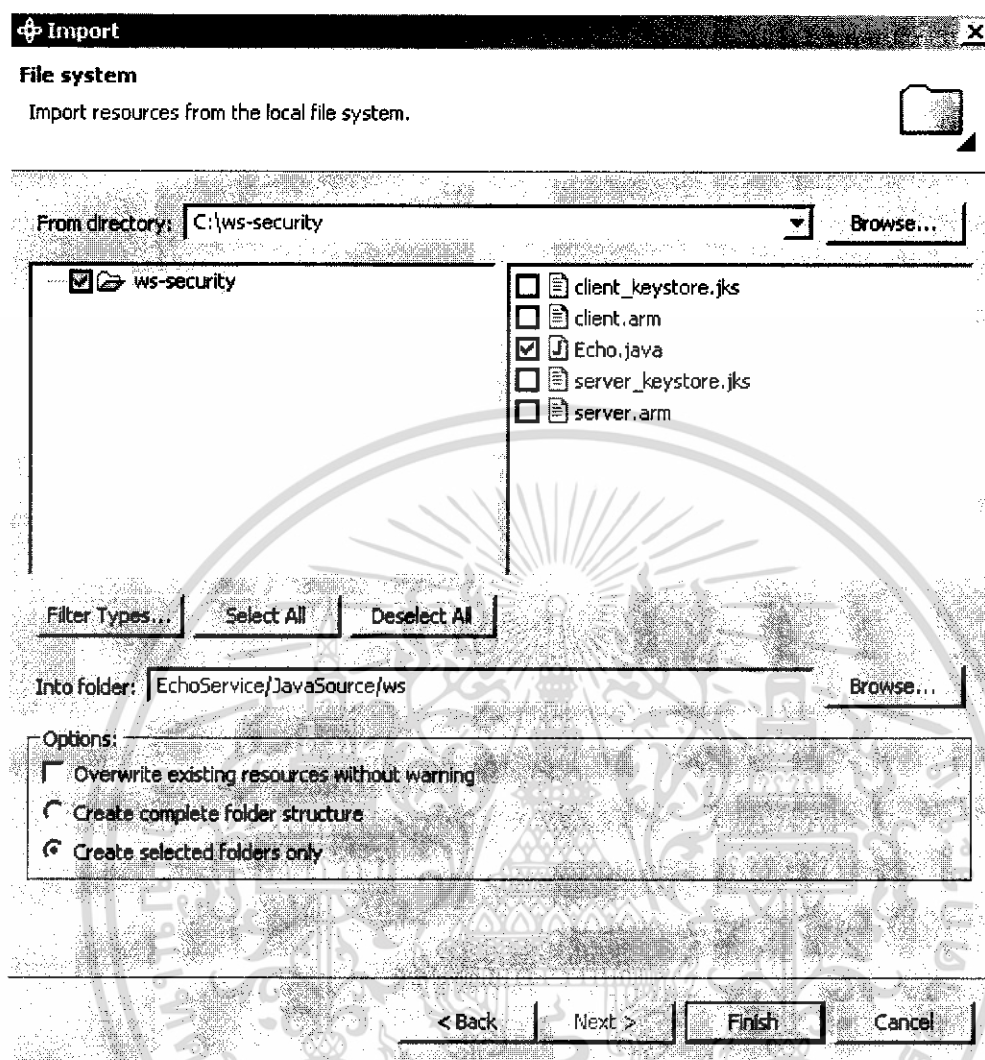


รูปที่ 4.19 แสดง การสร้างเว็บเซอร์วิส

2. เขียน function ที่ต้องการ provide เป็นเว็บเซอร์วิส โดยการสร้างไฟล์จาวาขึ้นมาใหม่ File->new ->Java ->Class เช่น

```
package ws;
public class Echo
{
    public String echo(String s)
    {
        return s;
    }
}
```

หรืออาจทำได้โดยการ Import ไฟล์จาวาที่มีอยู่แล้วโดยเลือก File->import เช่น



รูปที่ 4.20 แสดงการ import ไฟล์จาวา เพื่อใช้ในการทำ เว็บเซอร์วิส

3. ทำการสร้างเว็บเซอร์วิสโดยเลือก File->New ->other->Web Service->Web Service เลือก. Java bean Web Service, Overwrite files without warning checkboxes. Next: -> finish

Web Service

Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

Service

Web service type: Java bean Web Service

☒ Start Web service in Web project

☐ Test the Web service

☐ Launch the Web Services Explorer to publish this Web service to a UDDI Registry

☒ Generate a proxy

Client proxy

Client proxy type: Java proxy

☒ Test the generated proxy

☒ Overwrite files without warning

☒ Create folders when necessary

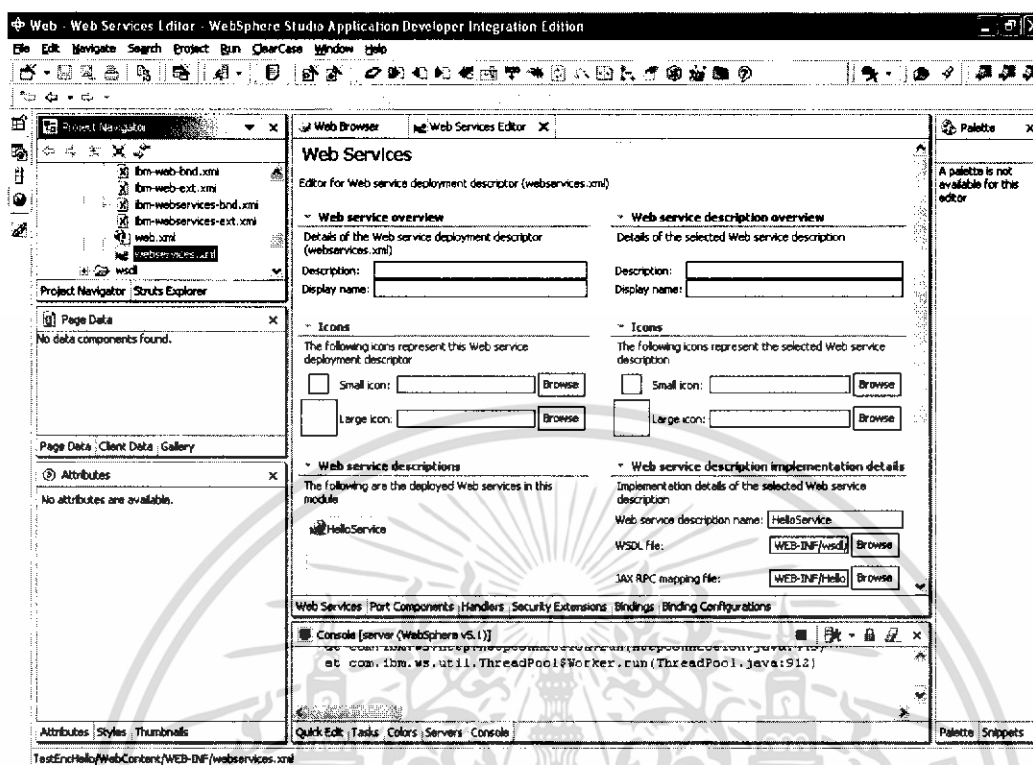
☐ Check out files without warning

< Back Next > Finish Cancel

รูปที่ 4.21 แสดงการสร้างเว็บเซอร์วิส

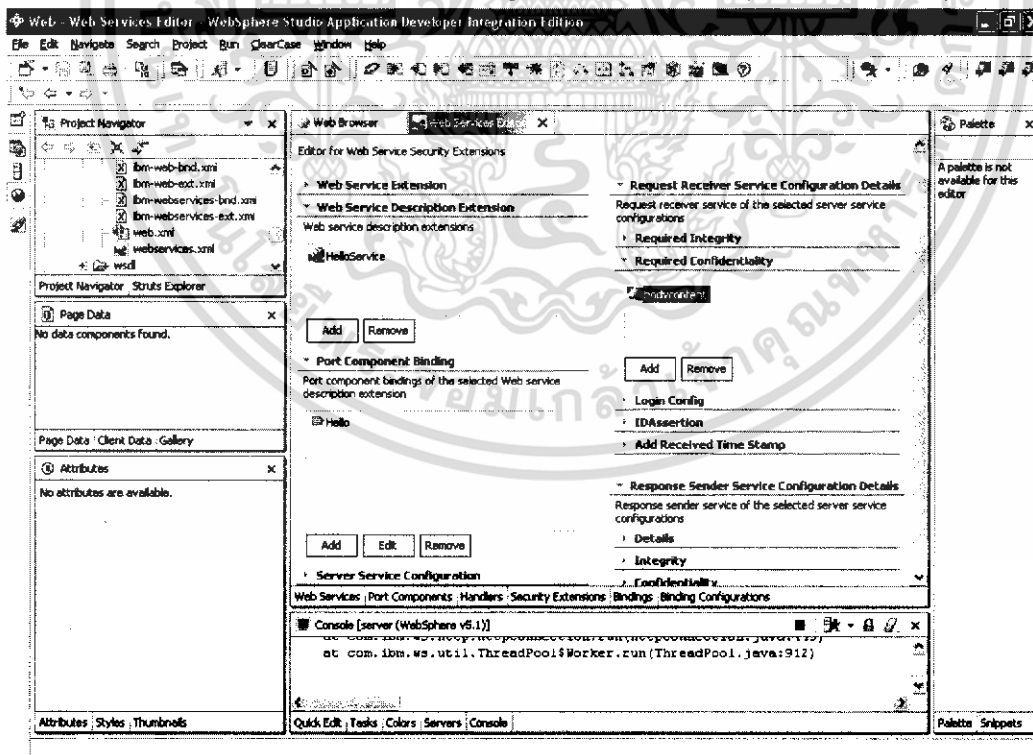
4.2.3.1.2. การ configure WS-Security แบบ Encryption บน Websphere

1. เข้าไปแก้ไขไฟล์ webservice.xml ซึ่งอยู่ใน directory -> WEB-INF -> webservice.xml



รูปที่ 4.22 แสดงข้อมูลในไฟล์ webservice.xml

2. เลือก Security Extension Tab -> Require Confidentiality -> Add -> bodycontent



รูปที่ 4.23 แสดงการConfigure ใน security extension

3. เลือก Binding Configurations TAB -> Request Receiver Binding Configuration-
 > Keylocator ระบุตามรูปด้านล่าง

Key locator dialog

Key locator name: Interops

Key locator class: com.ibm.wsspi.wssecurity.config.KeyStoreKeyLocator

☒ Use key store

Key store storepass: interops

Key store path: c:/Enc_key/interops-enc-receiver.jceks

Key store type: JCEKS

Key:

Alias	Key pass	Key name
interops-private-key	interops	interops

Add Remove

Property:

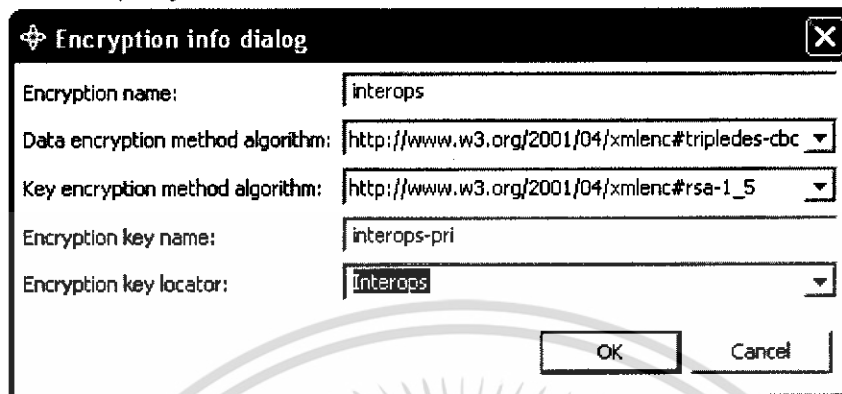
Name	Value
------	-------

Add Remove

OK Cancel

รูปที่ 4.24 แสดงการ configure Key ที่ใช้ในการ Encryption

4. เลือก Binding Configurations TAB -> Request Receiver Binding Configuration-
> Encryption และระบุตามรูป



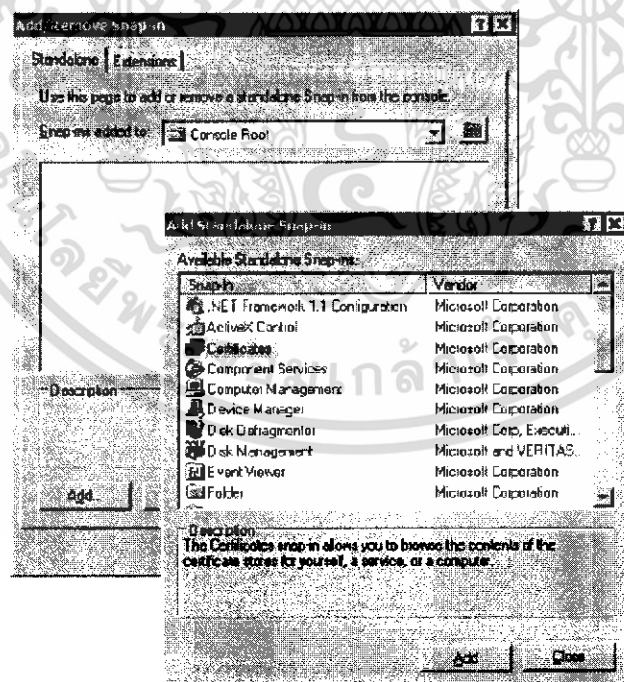
รูปที่ 4.25 แสดงการ configure encryption

จากนั้นกด Save และทำการ Restart Server และนี่คือการ configure WS-Security บน Websphere

4.2.3.2 การเซตค่าของ Microsoft Visual Studio .Net 2003 with WSE 1.0

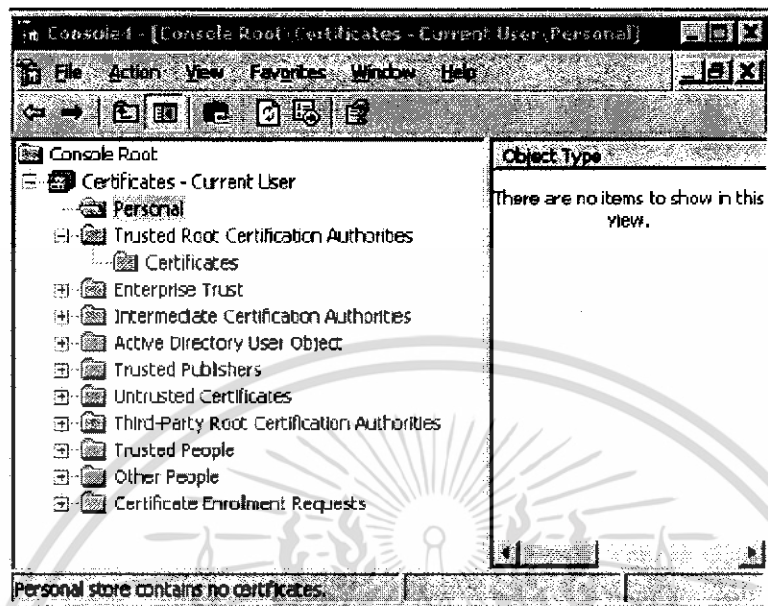
4.2.3.2.1 การ install key

1. ในการจะทำ Security บน Microsoft .Net จะต้องมีการ Add certificate ใส่ใน store ซึ่งสามารถทำได้โดย Start Menu -> run พิมพ์ mmc จากนั้นจะเข้าสู่ console ให้เลือก Add/Remove Snap-in ซึ่งจะแสดงหน้าจอตามรูปด้านล่างขึ้นมา



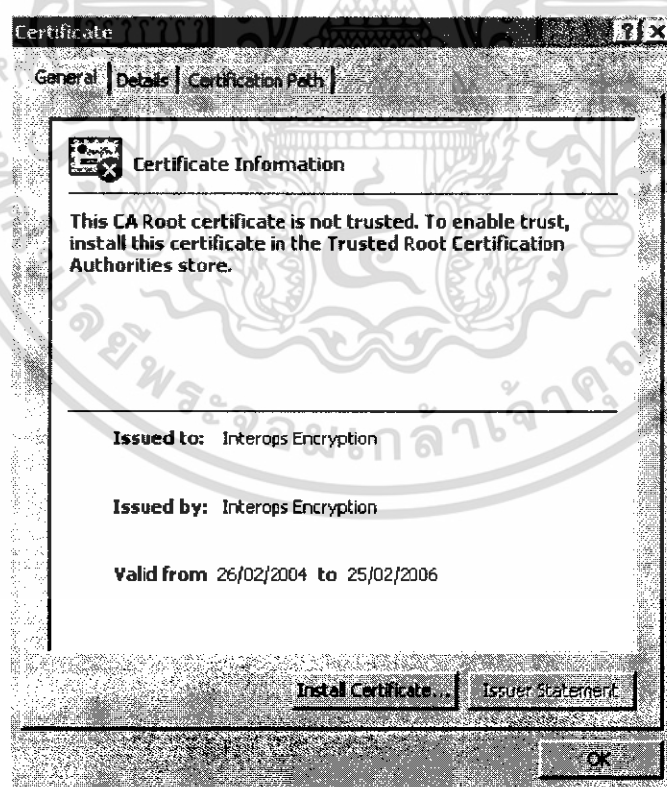
รูปที่ 4.26 แสดงการ เพิ่มคีย์

2.เลือก Certificate ->manage user account เมื่อเลือกแล้วจะแสดงหน้าต่างดังนี้



รูปที่ 4.27 แสดง Store ใน Windows

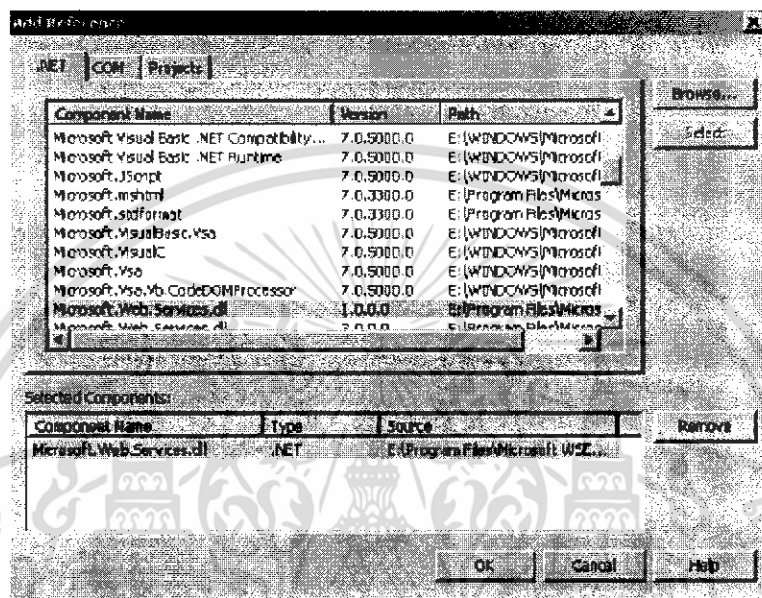
3. ต้องทำการ import public key เข้าไปใน Personal Store & Trusted Root Certificate Authority



รูปที่ 4.28 แสดง Certificate ที่ import เข้ามาแล้ว

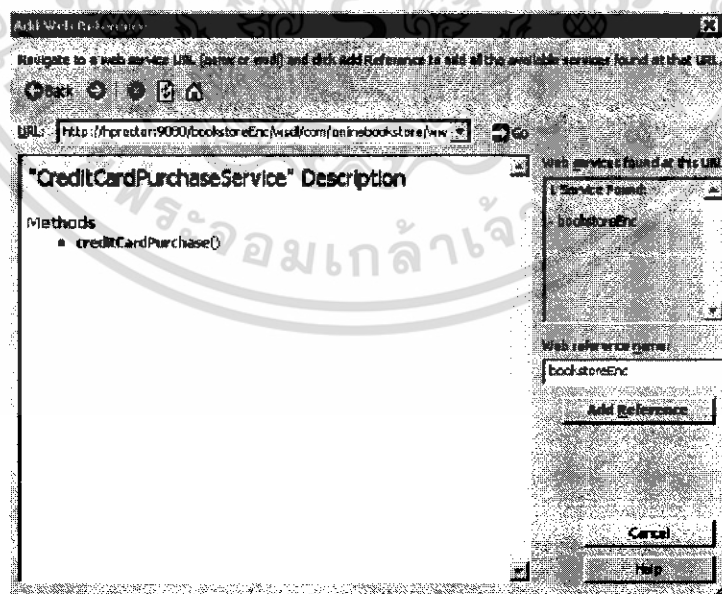
4.2.3.2.2. การสร้างเว็บแอปพลิเคชันของ ASP ด้วย Microsoft Visual Studio .Net 2003 ที่มี การติดตั้ง WSE 1.0 และ IIS เรียบร้อยแล้ว

1. ทำการสร้างเว็บเซอร์วิส โดยเข้าไปใน Visual Studio .Net 2003 จากนั้น เลือก ASP .Net Web Application เมื่อสร้างเสร็จแล้วจึงทำการ Add Reference โดย Add Microsoft.Web.Services.dll ซึ่งเป็นไฟล์ของ WSE 1.0



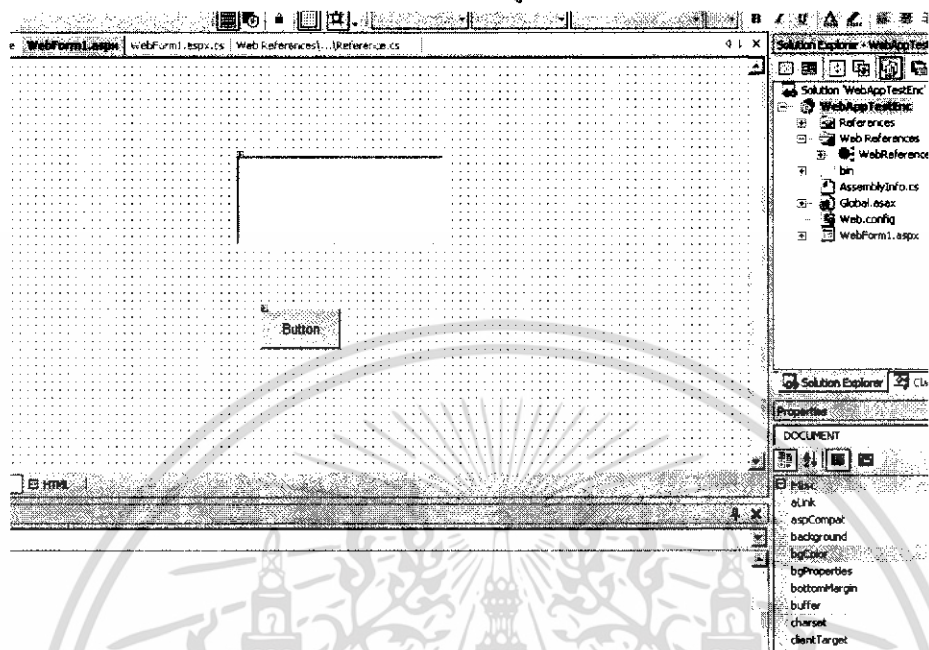
รูปที่ 4.29 แสดงการ Add reference Class ของ WSE 1.0

2. จากนั้นทำการ Add Web Reference ของ เว็บเซอร์วิส

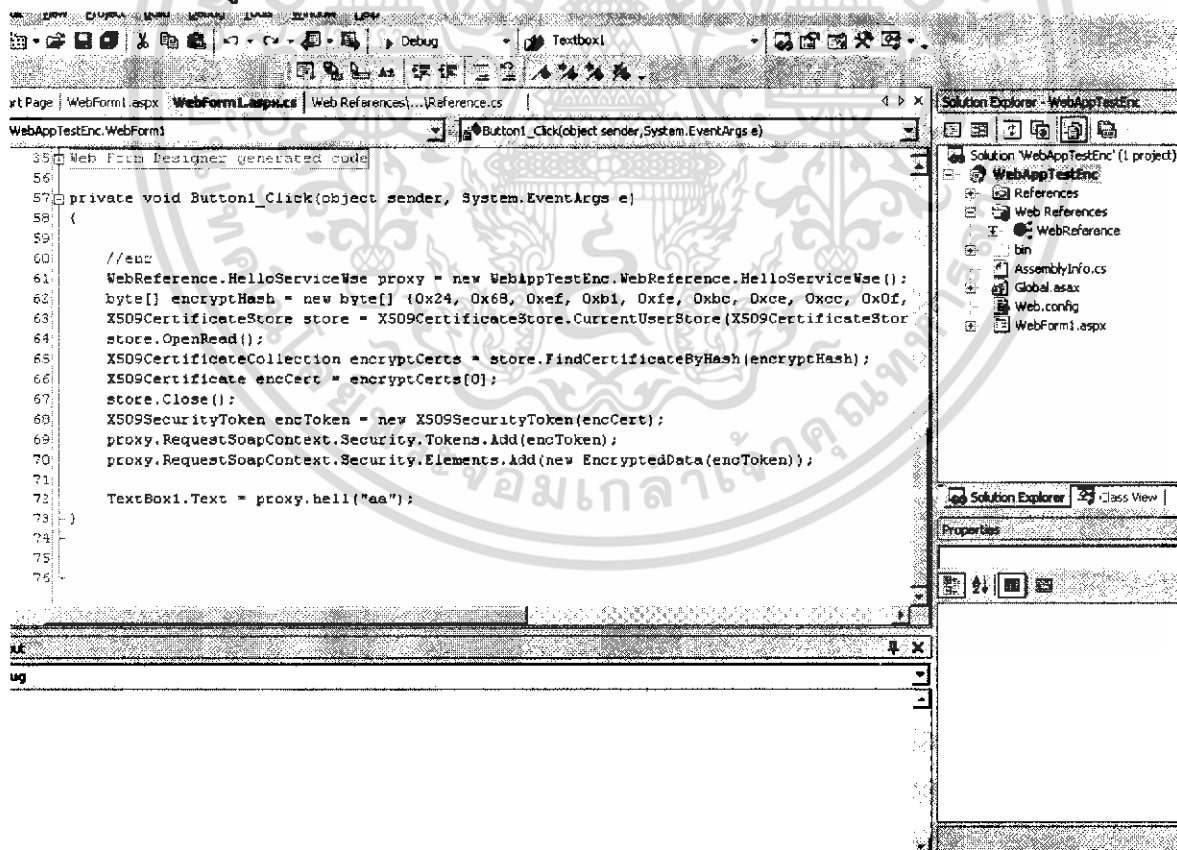


รูปที่ 4.30 แสดง Add Web reference ของ Web Service โดยระบุ WSDL

3. จากนั้นทำการ สร้างเว็บแล้ว เพิ่ม code ลงไปตามรูป



รูปที่ 4.31 ออกแบบหน้าตาของเว็บให้สอดคล้องกับเว็บเซอร์วิส

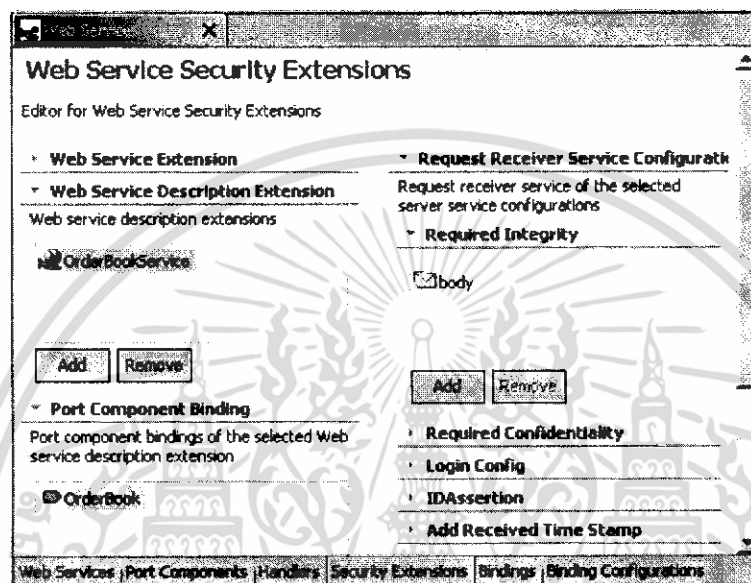


รูปที่ 4.32 เพิ่ม code ของโปรแกรมเพื่อใช้ในการ encryption

4.2.4. การเซตค่าเพื่อใช้ในการทำ WS-Security แบบ Digitalsignature ระหว่าง Websphere Application Developer 5.1 และ Microsoft Visual Studio .Net 2003

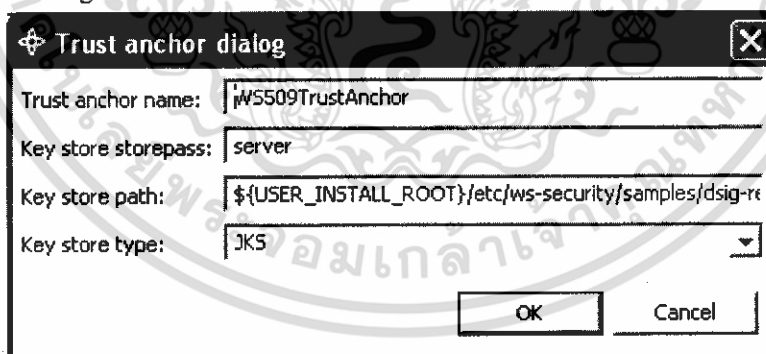
4.2.4.1. การเซตค่าของ Websphere Application Developer 5.1

1. เมื่อสร้างเว็บเซอร์วิสเสร็จแล้วให้ไปconfigure webservice.xml -> security Extension -> request Receiver Configuration -> require integrity->body



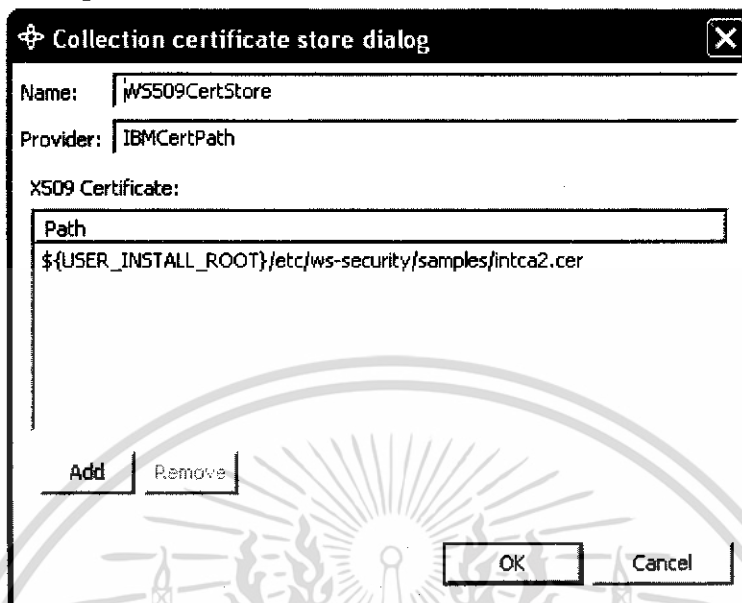
รูปที่ 4.34 แสดง require integrity

2. จากนั้นจึง Configure Binding configurations -> request receiver -> Trust Anchor เพื่อระบุคีย์ที่ต้องใช้ในการทำ signature



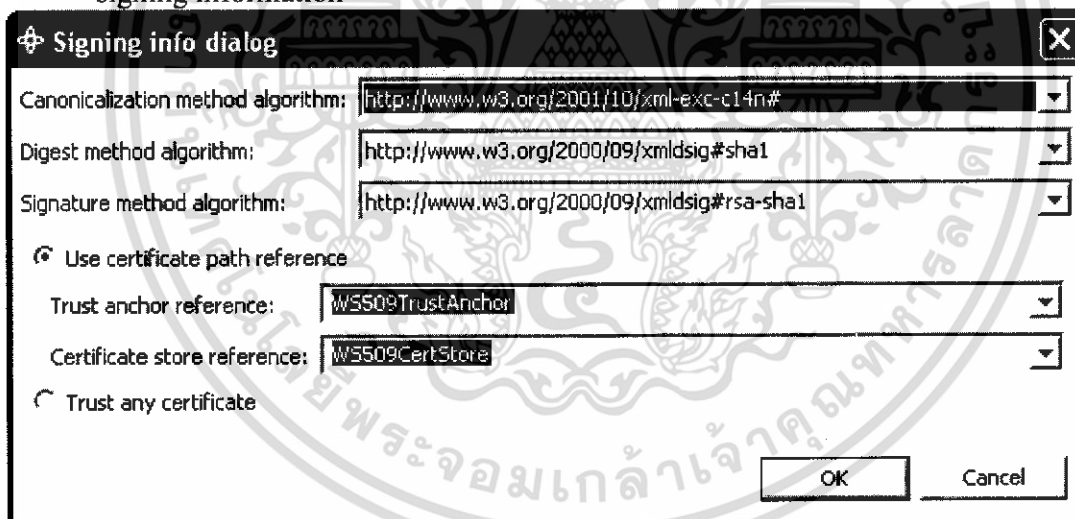
รูปที่ 4.35 แสดงการระบุคีย์ที่ต้องการใช้ใน signature

3. แล้วต่อมาจึง configure ส่วนของ certificate ใน certificate store list -> certificate store



รูปที่ 4.36 แสดง certificate ที่ใช้ trust

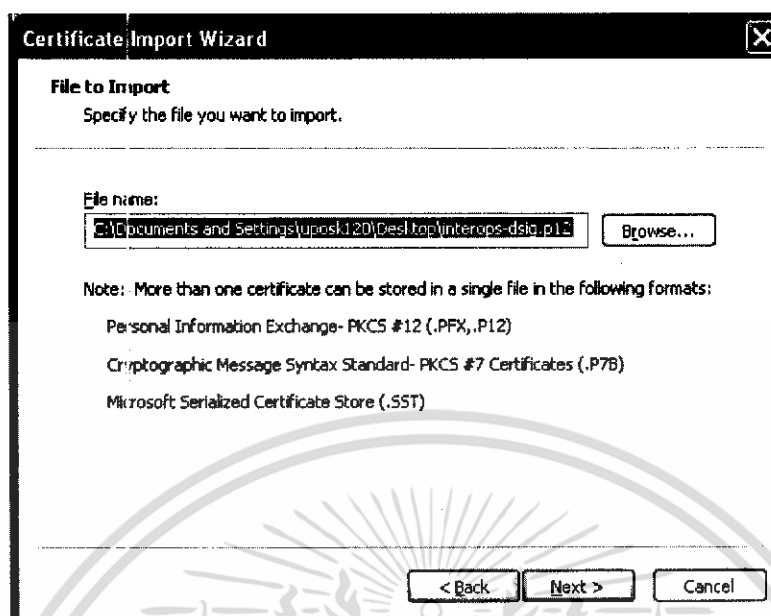
4. จากนั้นจึงไป configure เกี่ยวกับคีย์ ใน Binding configurations -> request receiver -> signing information



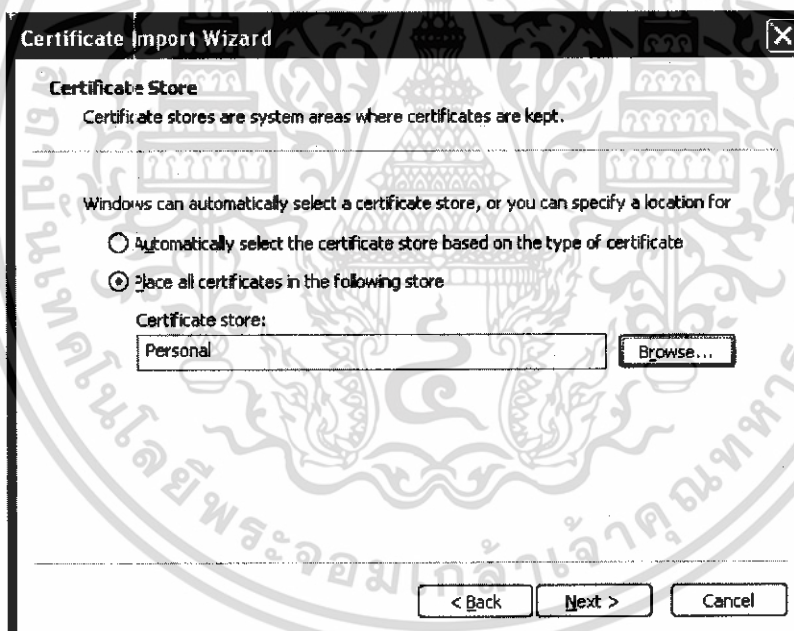
รูปที่ 4.37 แสดง การ configure ใน signing information

4.2.4.2. การเซตค่าของ Visual Studio .Net 2003

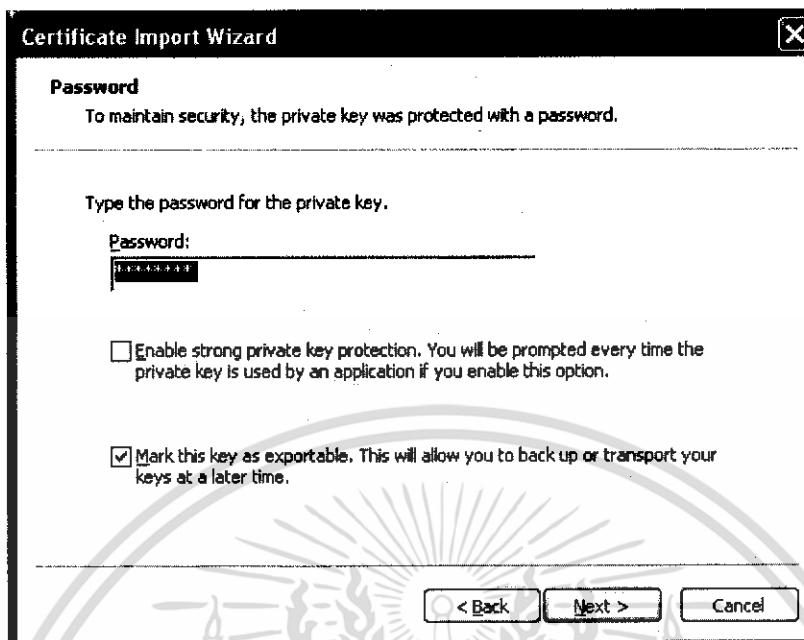
1. ต้องมีการ Add key สำหรับ Sign เพิ่มใน store โดยในที่นี้ ต้อง configure ในรูปแบบที่คล้ายๆกัน โดย ต้อง เพิ่ม คีย์ทั้ง 2 อย่าง คือ key ที่ใช้ในการ sign และ key ที่ใช้ในการ trust โดยในที่นี้ใช้ คีย์ ในที่นี้ใช้ คีย์ interops-root.cer ซึ่งเป็น public key ใน trust และ ใช้ คีย์ interops-dsig.p12 ซึ่งเป็น private key ใน personal โดยการ import เข้าไปเหมือนเดิม



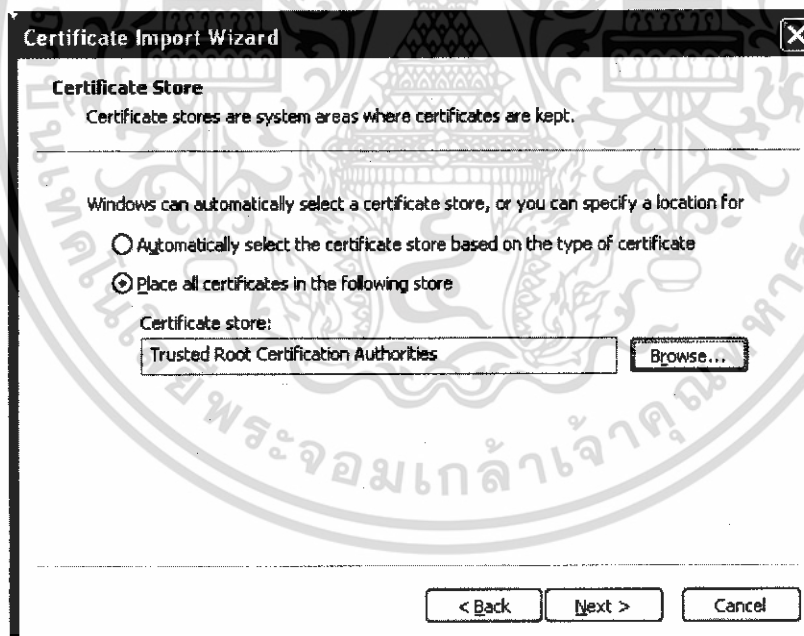
รูปที่ 4.38 แสดงการ import key interops-disg.p12



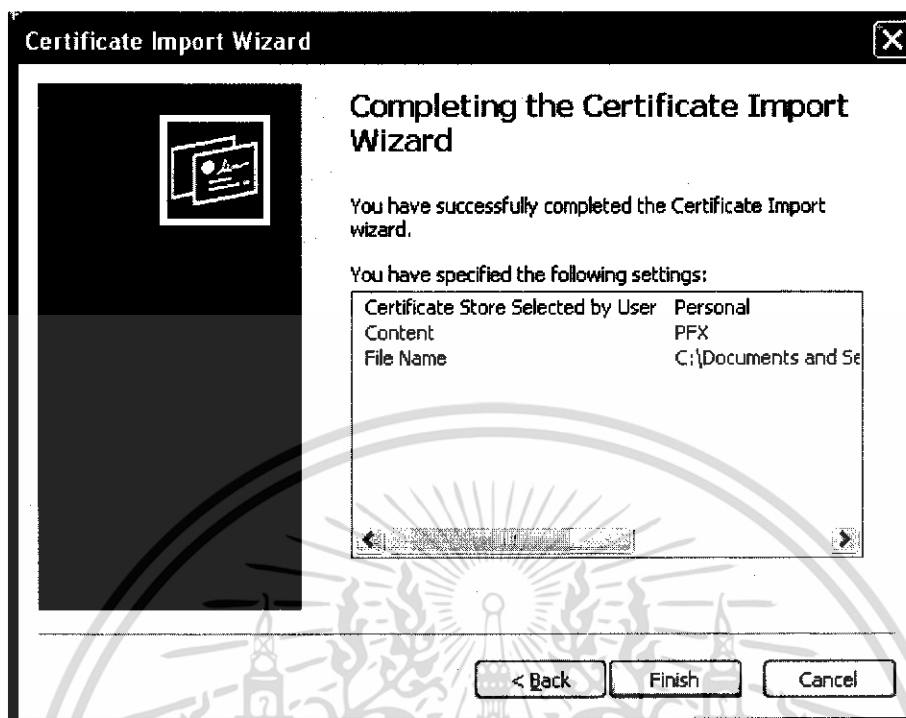
รูปที่ 4.39 แสดงผลการ import key ใน personal store



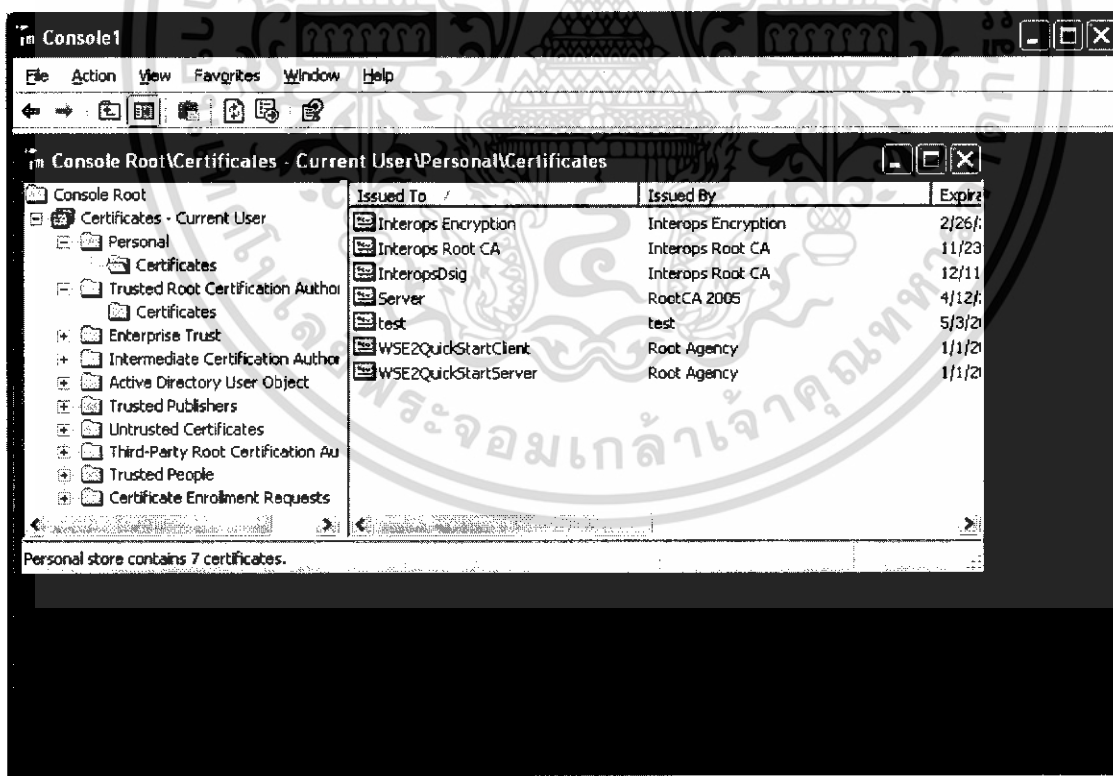
รูปที่ 4.40 แสดงการ import key pfx ที่มี private key ต้องมีการใส่รหัสผ่านด้วย



รูปที่ 4.41 ทั้ง 2 คีย์ต้องทำใส่ใน personal Store & Trust Root



รูปที่ 4.42 แสดงผลการ import ทั้ง 2 สำเร็จ



รูปที่ 4.43 แสดง key และ certificate ที่ import เข้าไปแล้ว

2. ต้องมีการเพิ่ม code เข้าไปเพื่อให้สามารถใช้digital signature ได้ ใน Microsoft .net ดังนี้

```

private void Button1_Click(object sender, System.EventArgs e)
{
    encRef.HelloServiceUse proxy = new WebAppTestCer2.encRef.HelloServiceUse();
    byte[] encryptHash = new byte[] {0x24, 0x68, 0xef, 0xb1, 0xfe, 0xbc, 0xce, 0xcc, 0x0f};
    byte[] signHash = new byte[] {0x60, 0xac, 0x1e, 0x05, 0x5c, 0x84, 0xeb, 0xf2, 0x92, 0x};
    X509CertificateStore store = X509CertificateStore.LocalMachineStore(X509CertificateStore.OpenRead());
    X509CertificateCollection encryptCerts = store.FindCertificateByHash(encryptHash);
    X509CertificateCollection singCerts = store.FindCertificateByHash(signHash);
    X509Certificate encCert = encryptCerts[0];
    X509Certificate signCert = singCerts[0];
    (local variable) Microsoft.Web.Services2.Security.X509.X509CertificateCollect

    X509SecurityToken encToken = new X509SecurityToken(encCert);
    X509SecurityToken signToken = new X509SecurityToken(signCert);

    proxy.RequestSoapContext.Security.Tokens.Add(encToken);
    proxy.RequestSoapContext.Security.Tokens.Add(signToken);
    proxy.RequestSoapContext.Security.Elements.Add(new EncryptedData(encToken));
    proxy.RequestSoapContext.Security.Elements.Add(new MessageSignature(signToken));
    store.Close();

    TextBox1.Text = proxy.hell();
}

```

รูปที่ 4.44 แสดง code ที่เพิ่มเข้าไปให้สามารถทำได้ ทั้งDigital signature & Encryption

```

1 <soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ><soapenv:Header><wsse:Security
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext" soapenv:mustUnderstand="1">
2   <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
3     <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/></EncryptionMethod>
4     <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
5       <wsse:SecurityTokenReference>
6         <wsse:KeyIdentifier>X/cjNKG2FMeR6VQ3GZTAskf601-/</wsse:KeyIdentifier>
7       </wsse:SecurityTokenReference>
8     </KeyInfo>
9     <CipherData>
10      <CipherValue>n40IN6m/IKuK6RRIP6AepDFs+IZ02v0PTYrj7PLOCv3VghvleKqwx7fINDWwGqN/MUVKMH-
        ySOVB5WVG7vA+7E69j6DP7nYIZHycBe2ZthSvYRTO6MbJROw7H59+mqB7OJyDjzIQ7dAxmuHjrar7R9qN8p2d3M/H5gAhns=</CipherValue>
11    </CipherData>
12    <ReferenceList>
13      <DataReference URI="#wsse:security_encryption_id_219524811905120095"/></DataReference>
14    </ReferenceList>
15  </EncryptedKey>
16  <wsse:BinarySecurityToken xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility" EncodingType="wsse:Base64Binary"
    Value="http://www.w3.org/2003/06/utility" wsu:Id="#wsse:security_binary_security_token_id_5637437789421640683">
17    MIIDOTCCAAGAwIBAgICAQOWDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCIAxETAP
18    BgNVBAGlTCETbmlmFnYxdhMQwwCgYDVQQKEwNjQ0K0xDDAkgBgNVBAStA1RSTDEQMA4GA1UE
19    AxiHHSW50IENBMjAeFw0wMTEwMDEwOTU0MDZaFw0wMTEwMDEwOTU0MDZaMFQxCzAJBgNV
20    BAYTAkRlQWwvDQYDVQQGEWhLYW5hZ2F3YTEuMMAoGA1UEChMDSUJNMQwwCgYDVQQLEwNU
21    UkwxFjAUBGNVBAMTD0VNPQVBSZXF1ZXN0ZXIwZ28wDQYJKoZIhvcNAQEBBQADgY0AMIGJ
22    AoGBAMy3PiZ1mPhrEsBvYiQuilPV3Uis5Yy6hmxc2YwYc2nNDBPzkshWUJ/O+fk+DNdY
23    6KEHmuDreVcme48J9X1a5avRlksQIKpK0Vn4eBys2/vkwyZQhDaFji79/MvNTRW8E
24    Vy99FNKw4PFhOae1t0CnBulH/lluGoz9EITV+IqMBAAAGggEmMIBjAJBgNVHRME
25    AJAAMAsGA1UdDwQEAwIF4DAsBgqhkgBhvCAQ0EHxYdT3BibINTTCBHZW5lcmlmF0ZWQg
26    Q2VydGlmZWwvDQYDVVR0BBYEFwF3FD1cXk4j4zwlGpA4cuOAZ4IMIG6BGNVHSMG
27    gblwge+AF1351NU4+WRy09val9zOsP7Qv09voYGSplGPMIGMMQswCQYDVQQGEwKUDER
28    MA8GA1UECBMIS2FyYWhhd2E5X2ZANBgNVBAcTBllhbwF0bzMMAoGA1UEChMDSUJNMQww
29    CgYDVQQLEwNUUkwxFjAUBGNVBAMTEFNQVAgM4xIFRlc3QgO0ExIjAgBgkqhkiG9w0B
30    CQSWZ21hcnV5YW1hQGpwLmlibS5jb2Z2CgEBA0GCSqGSIb3DQOEBSBUAA4GBAHkthdGD
31    gCvdlL5U/XUo74XpOGd/r1owBmMdb1TYvDQyzwbOHC7IKUlnKk7SotwSLSDUP571i
32    iMXu3IRd14YCoDMMFUDXh9V7212luXccx0s1S5KN0D3xW97LLNegOC0/b+aFD8XKw2U
33    5ZwbmFTRrs097m+09RosDKKLIM
34  </wsse:BinarySecurityToken>
35  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
36    <SignedInfo>
37      <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc14#"/></CanonicalizationMethod>
38      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/></SignatureMethod>
39      <Reference URI="#wsse:security_binary_id_5637437789421640683"/>
40    </SignedInfo>
41    <Transform>
42      <Transform Algorithm="http://www.w3.org/2001/10/xml-exc14#"/>
43      <InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc14#" PrefixList="ec:soapenc:wsd:default:wsu:soapenv"/>
44    </Transform>
45    </Transform>
46    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/></DigestMethod>
47    <DigestValue>3f4HlgszYd7teVW37Jot08mc=</DigestValue>
48  </Signature>
49  <SignatureValue>
50    Ys1AczkU52JE7J/DxP2u0VXBxEhcQl6sh3eFpeQNV6QdJuHMBv0r2QFO
51    zyWTWghqQ8F4xcYjlyHg8dsfWDchp8JwW5Lc/RyUavJ6vAzPj7N3dCD/CxSG
52    TZuqh9J4xbGzJeTAqoYbpS30FXG UgnPCjJqk0sgsisjJIF610=
53  </SignatureValue>
54  <KeyInfo>
55    <wsse:SecurityTokenReference>
56      <wsse:Reference URI="#wsse:security_binary_security_token_id_5637437789421640683"/></wsse:Reference>
57    </wsse:SecurityTokenReference>
58  </KeyInfo>
59  </wsse:BinarySecurityToken></wsse:Security></soapenv:Header><soapenv:Body>
  <xs:element base="http://schemas.xmlsoap.org/ws/2003/06/utility"
    id="wsse:security_encryption_id_219524811905120095" Type="http://www.w3.org/2001/04/xmlenc#Content"/><EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
60  </EncryptionMethod>
61  <CipherData>
62    <CipherValue>29VrQiaHeqYT7fWMDosZwQgptxBEal7VgnQltnN7IAsamPUSbs4TQ==</CipherValue>
63  </CipherData>
64  </EncryptedData>
65  </soapenv:Body></soapenv:Envelope>

```

Keyinfo&
Encryptkey

Binary
Security
Token

Security
Algorithm

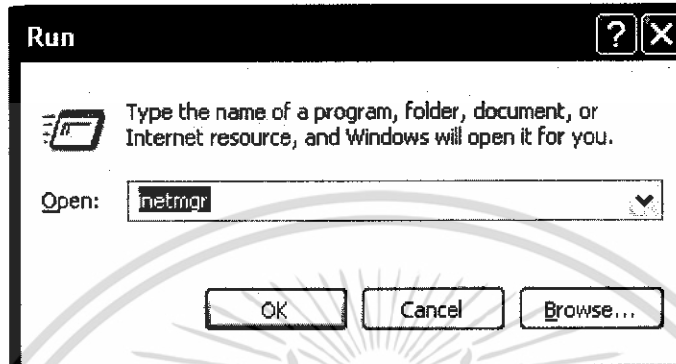
Signature
Value

SOAP
Body&
CipherData

4.2.5. SSL

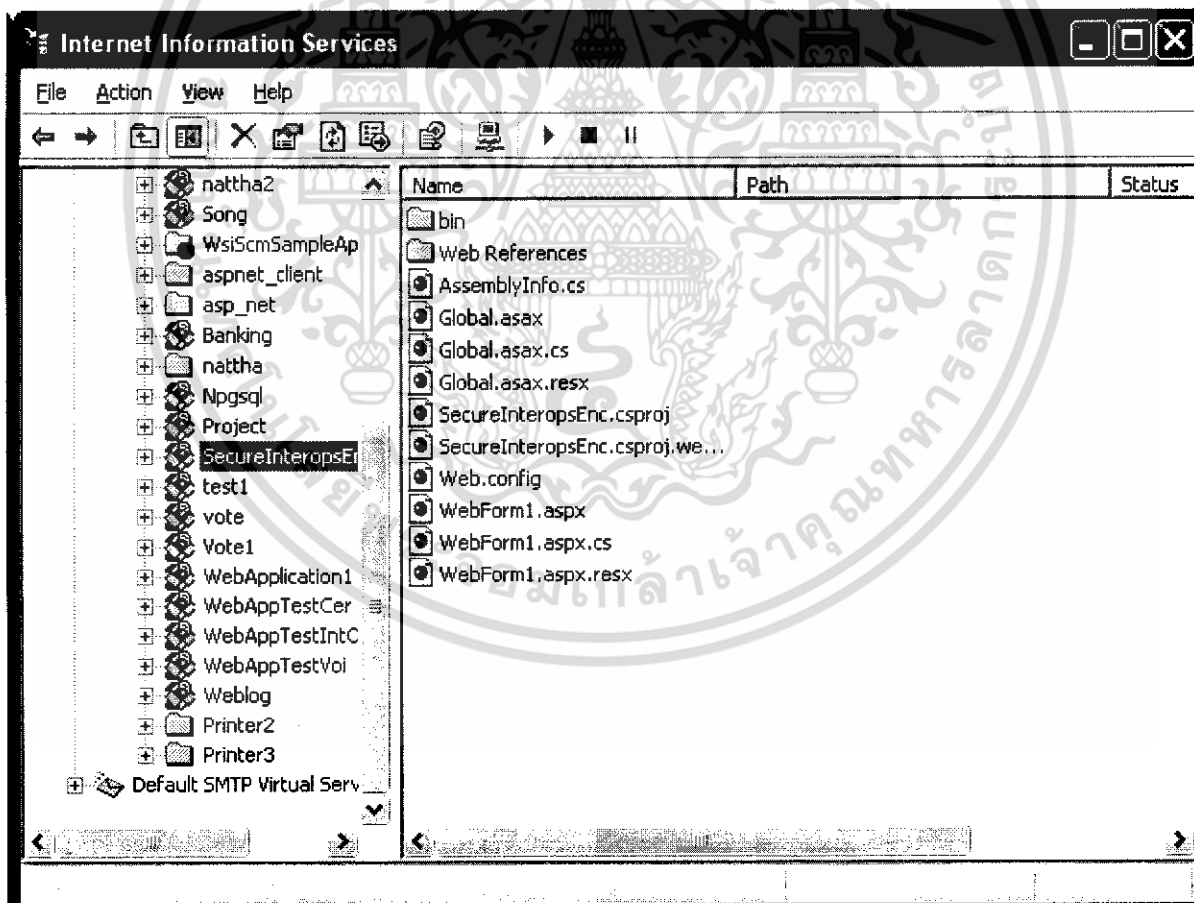
4.2.5.1 การติดตั้ง SSL บน IIS สำหรับใช้ใน ASP.Net Web Application

1. จัดการในส่วนของ IIS Manager โดย พิมพ์ inetmgr ที่ run...



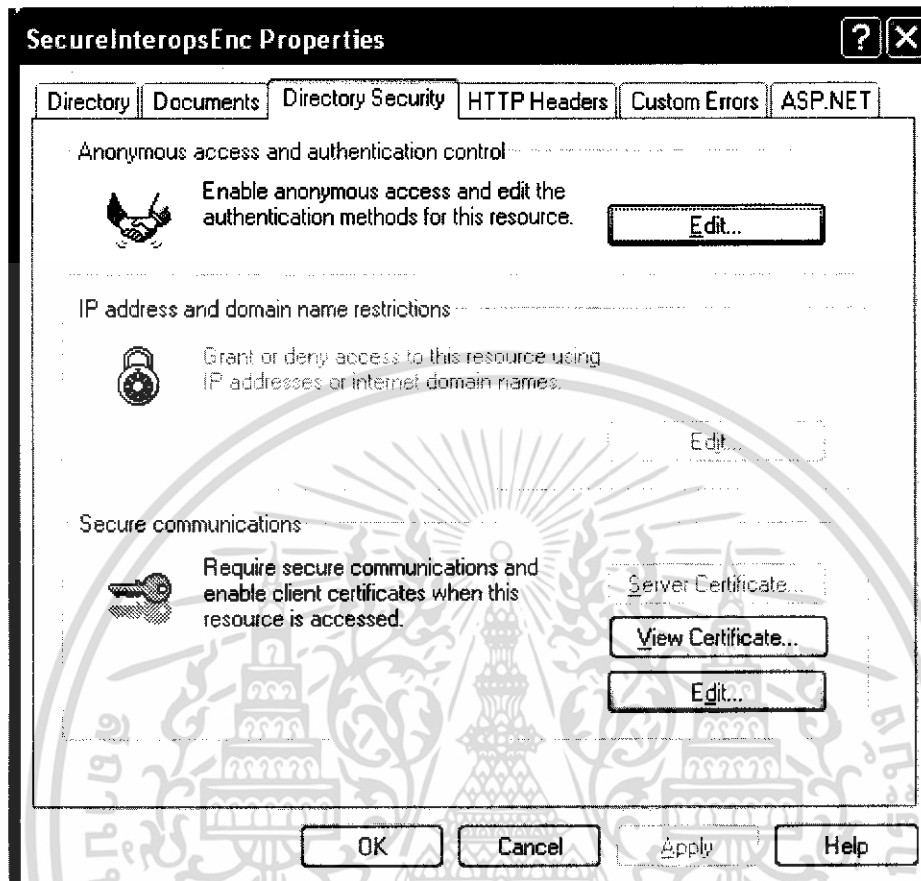
รูปที่ 4.46 แสดง การเข้าไปจัดการใน IIS Manager

2. เลือกเว็บไซต์ที่ต้องการทำ SSL



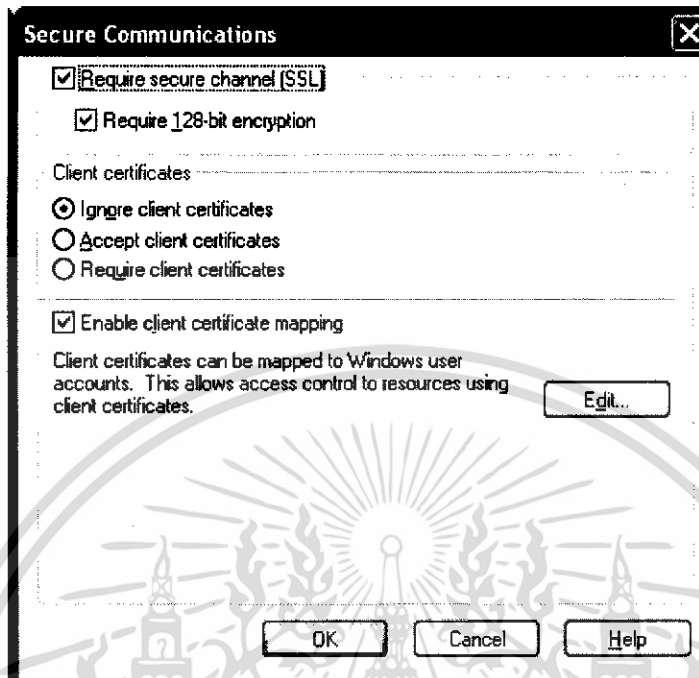
รูปที่ 4.47 แสดงการเลือกเว็บไซต์

3. เมื่อเลือกเว็บไซต์ได้แล้ว ให้ทำการ Configure secure communication โดยเลือก Edit



รูปที่ 4.48 แสดง properties ในเว็บต่างๆ

4. เลือก Require SSL /Require 128-bits encryption / Enable client certificate mapping

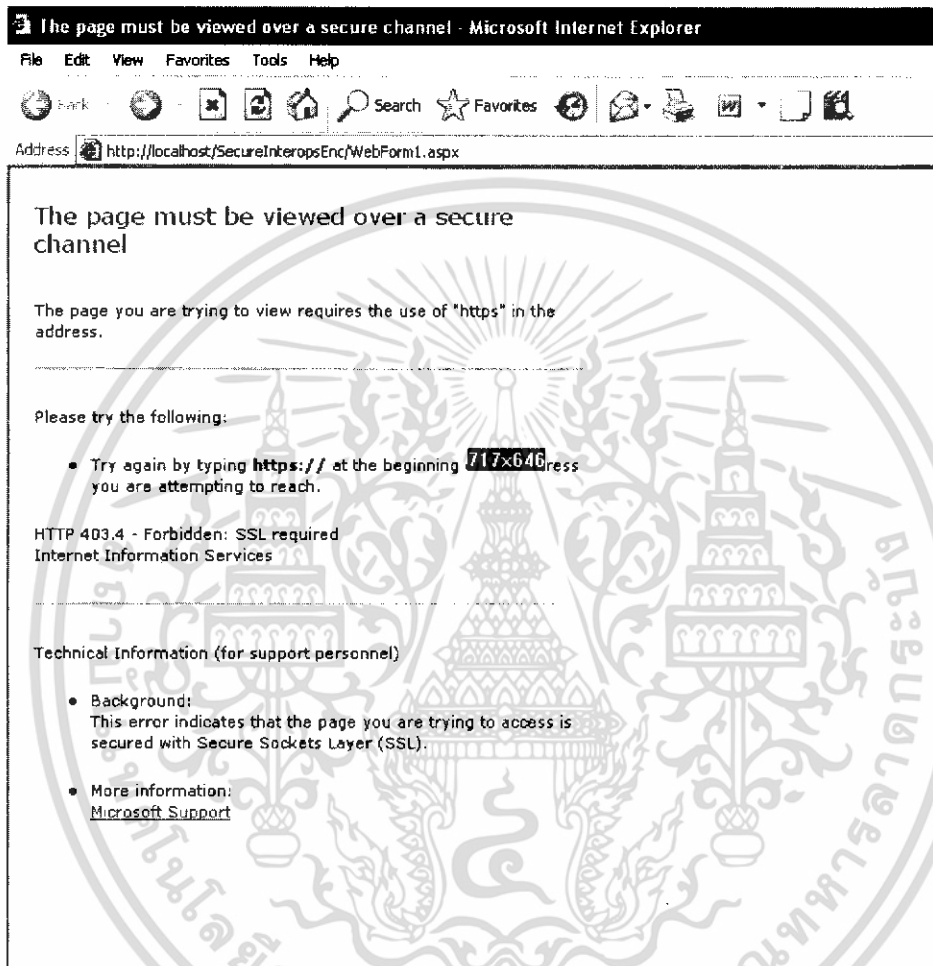


รูปที่ 4.49 แสดงการ ติดตั้ง SSL

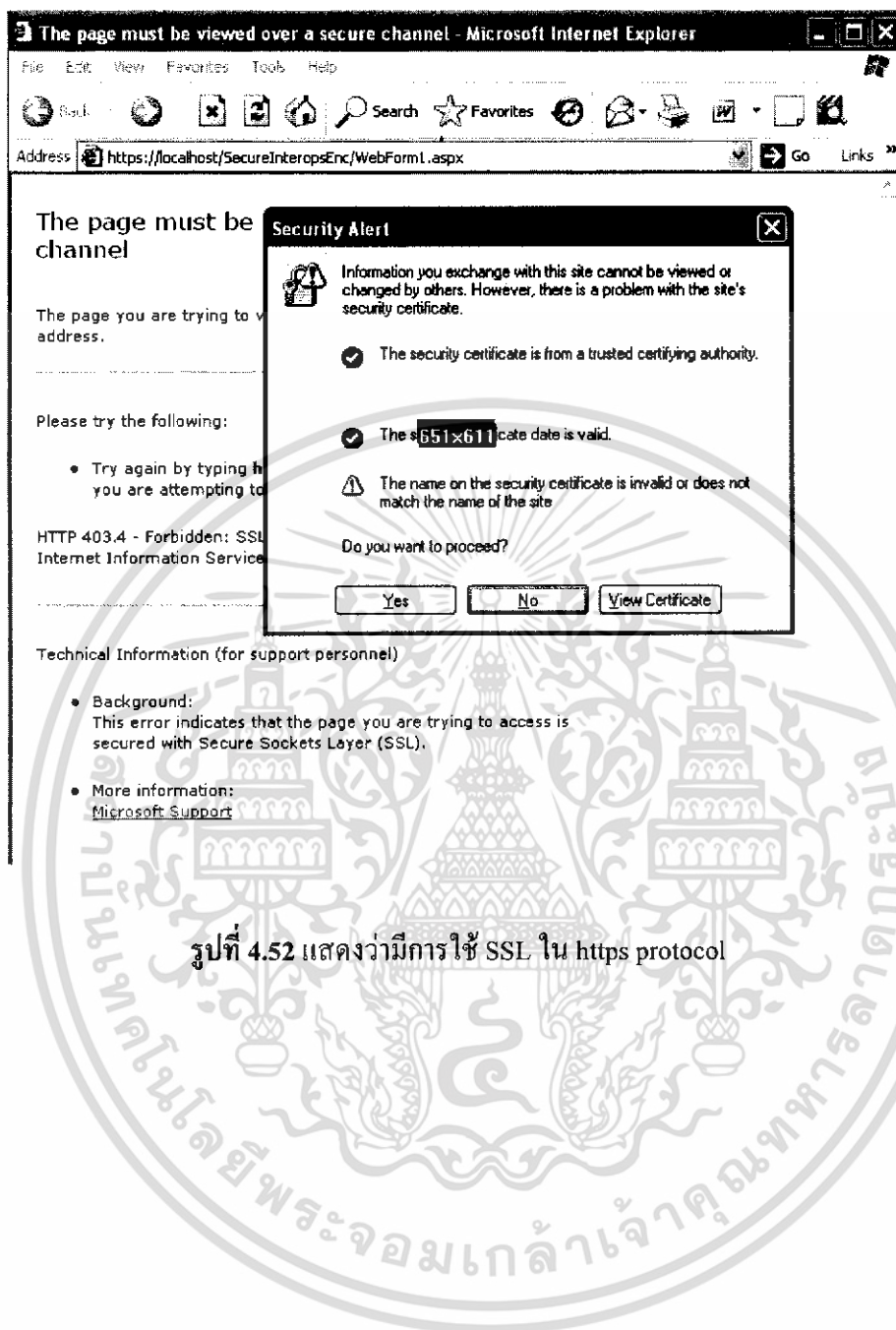


รูปที่ 4.50 แสดง certification ที่จะนำไปใช้ในการ SSL

เมื่อมีการ Configure SSL ในเว็บแล้ว จะไม่สามารถเรียกใช้เว็บผ่าน http protocol ได้ ต้องเรียกใช้งานในรูปแบบของ SSL เท่านั้น คือ https protocol จากนั้น จะมี ข้อความเตือนเรื่อง certificate เมื่อกดตกลงแล้ว จึงเป็นการใช้ SSL



รูปที่ 4.51 แสดงให้เห็นว่าเมื่อมีการ configure SSL แล้วจะไม่สามารถเข้าเว็บผ่าน http protocol ได้



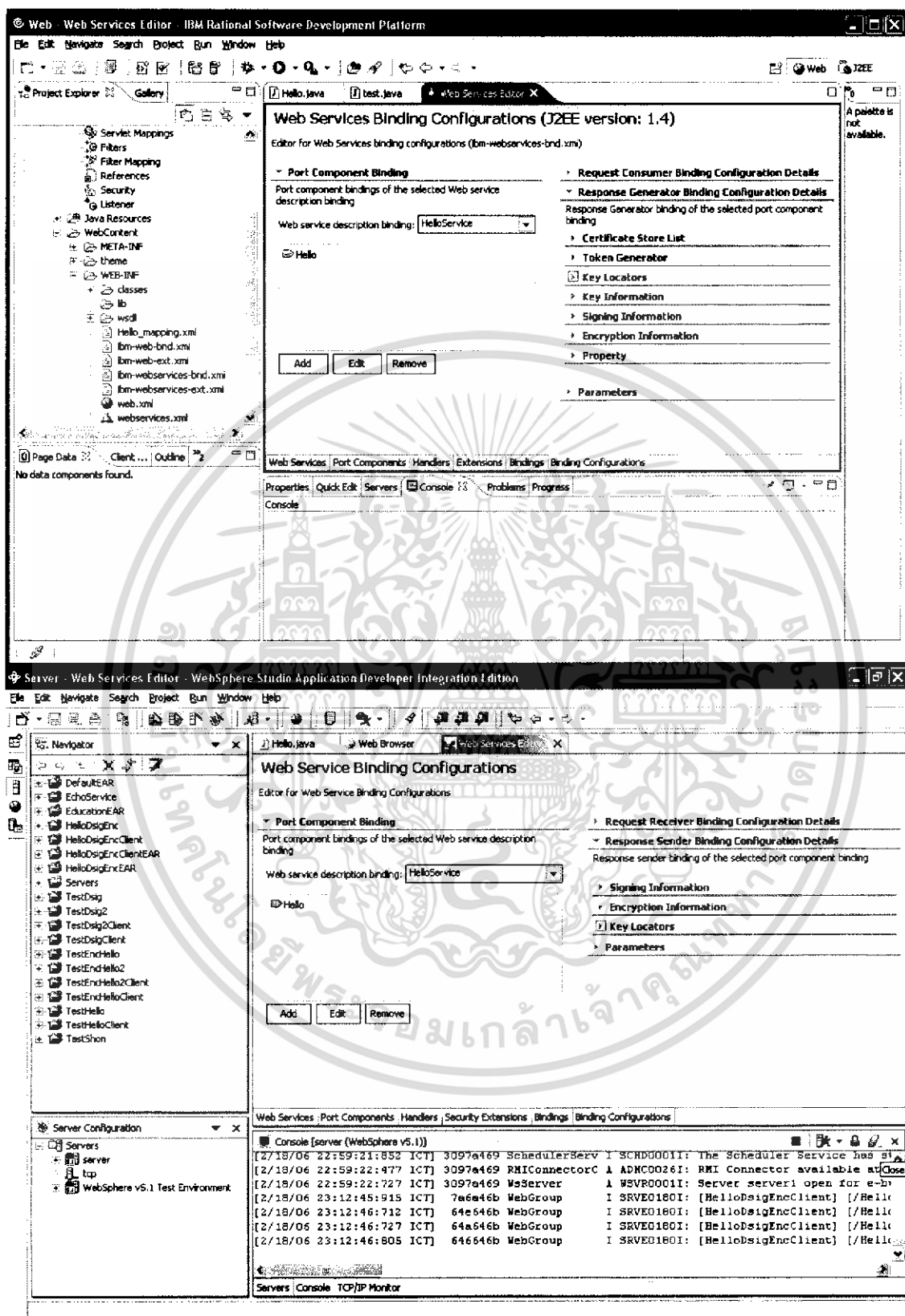
รูปที่ 4.52 แสดงว่ามีการใช้ SSL ใน https protocol

4.3. การทดลองWS-Security กับ เครื่องมืออื่นๆ ดังนี้

1. Websphere Application Server 6.0 กับ Microsoft Visual Studio 2003 with WSE 2.0
2. Websphere Application Developer 5.1 กับ Microsoft Visual Studio 2003 with WSE 2.0
3. Rational Application Developer 6.0 กับ Microsoft Visual Studio 2003 with WSE 2.0
4. Eclipse 3.1 with Apache Tomcat กับ Microsoft Visual Studio 2003

จากผลการทดลองสามารถสรุปรูปแบบของ WS-Security ของแต่ละระบบได้ดังนี้

1. หากไม่มีการใช้ WS-Security สามารถส่งข้อมูลถึงเว็บเซอร์วิสได้หมด
2. หากนำ WS-Security มาใช้ จำเป็นต้องทำให้การ Encryption ตรงตามมาตรฐานของ WS-Security 2004 ซึ่งระบุไว้ใน OASIS ซึ่งมีความละเอียดอ่อนในการระบุชนิดของข้อมูลใน element ต่างๆ โดยต้องระบุตรงกันทั้ง Client และ Server
3. สิ่งสำคัญที่จำเป็นต้องใช้คือ Key และ Certificate ที่สร้างขึ้นมาเพื่อใช้ในการจัดการเรื่องความปลอดภัยบนเว็บเซอร์วิสต้องทำให้ สามารถนำไปใช้ได้ ทั้งใน Java และ Microsoft .Net
4. ถึงแม้จะมีหลุดออกมาให้ทดลองมากมายแต่ทุกที่ นำไปใช้ได้จริงๆ และสามารถข้ามแพลตฟอร์มระหว่าง .Net และ Java ได้ นั้นมีน้อยมาก ซึ่งในขณะนี้ tool ชนิดที่พร้อม และรองรับ WS-Security 2004 ได้แก่ Rational Websphere Application Developer 6.0 และ Microsoft Visual Studio 2003 with WSE 2.0 และ Microsoft Visual Studio 2005 with WSE 3.0 ซึ่งทุกเหล่านี้นี้มีความต้องการทรัพยากรเครื่องในระดับที่สูงพอสมควร อีกทั้งรูปแบบการ configure ที่แตกต่างกันทำให้เกิดการสับสนอย่างมาก
5. Microsoft .Net ค่อนข้างเสถียร ซึ่งเมื่อมีการเปลี่ยนแปลงหรือแก้ไข Security แทนจะไม่ต้องแก้ไข code หรือโปรแกรมเลย ต่างจาก Java หากมีการเปลี่ยนแปลงอาจต้องสร้างโปรเจคใหม่จึงจะรันได้



รูปที่ 4.53 จะเห็นว่า Rational & Websphere มีส่วนคล้ายกันมาก แต่สนับสนุน Java ต่างเวอร์ชันกัน

Token Generator dialog

Token generator name: gen_sightgen

Token generator class: com.ibm.wsspi.wsssecurity.token.X509TokenGenerator

Security token: [dropdown]

☒ Use value type

Value type: gen_sightgen_vtype

Local name: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509

URI: [empty]

Call back handler: com.ibm.wsspi.wsssecurity.auth.callback.X509CallbackHandler

User ID: [empty]

Password: [empty]

☒ Use key store

Key store storepass: server

Key store path: \${USER_INSTALL_ROOT}/etc/ws-security/samples/dsig-receiver.ks

Key store type: JKS

Key:

Alias	Key pass	Key name
soapprovider	server	CN=SOAPProvider, OU=TRL, O...

Add Remove

Call back handler property:

Name	Value
------	-------

Add Remove

Property:

Name	Value
------	-------

Add Remove

OK Cancel

รูปที่ 4.54 ส่วนที่เพิ่มเข้ามาใน Rational 6.0 คือ Token Generator ซึ่งไม่มีในชุดอื่นๆ

4.5. เปรียบเทียบการทำงานของเว็บเซอร์วิสจาวาและไมโครซอฟต์ดอทเน็ต

ได้มีการทดสอบประสิทธิภาพการทำงานของ Web Service J2EE และ Web Service Microsoft.Net โดยบริษัท Sun Microsystems ซึ่งผลที่ได้ปรากฏว่า Microsoft .NET สามารถทำคะแนนได้ดีกว่า โดย จาวาเองก็โดดเด่นในบางเรื่องด้วยเช่นกัน จาวามีชุดเอพีไอที่ใช้จัดการ XML ได้ โดยตรง สนับสนุนการใช้งานร่วมกับ SOAP, UDDI และ WSDL แต่งานส่วนใหญ่ยังต้องเขียนขึ้นเอง ซึ่งเป็นงานที่ต้องใช้เวลาพอสมควร ในขณะที่การพัฒนาเว็บเซอร์วิสบน Microsoft.Net นั้นมีเครื่องมือหลายอย่างที่จะช่วยในการทำงาน ลดเวลาที่ต้องใช้ในการทำงานและทำได้ง่ายมากกว่า J2EE พอสมควร

ตามที่กล่าวข้างต้นแล้ว J2EE มีจุดเด่นที่สามารถทำงานได้บนฮาร์ดแวร์และระบบปฏิบัติการทุกชนิด ไม่ว่าจะเป็นวินโดวส์ ยูนิกซ์ ลินุกซ์ พืชี จนถึง เมนเฟรม เพียงแต่ต้องมี Java Runtime Environment บนแพลตฟอร์มเท่านั้น ดังนั้นเรื่องพอร์ตเทบิลิตี้ของ J2EE ก็เป็นเรื่องที่เป็นไปได้จริง แต่ในทางปฏิบัติก็ยังมีจุดอ่อนที่อาจทำให้เกิดปัญหาได้ เนื่องจาก J2EE เป็นเพียงมาตรฐานที่ Sun ออกแบบขึ้นมาเท่านั้น หากผู้พัฒนาซอฟต์แวร์ไม่ว่าจะเป็น IBM, Oracle, BEA หรือ Sun เอง ไม่ได้ปฏิบัติตามมาตรฐานอย่างเคร่งครัด พอร์ตเทบิลิตี้ที่เราเคยคิดว่ามีก็อาจไม่เป็นอย่างที่คาดก็ได้

ถ้าพิจารณา Microsoft.Net เนื่องจาก .Net ทำงานบนวินโดวส์ ดังนั้นจึงไม่มีปัญหาเรื่องการสนับสนุนฮาร์ดแวร์ แต่การทำงานบนแพลตฟอร์มอื่น ปัจจุบันยังเป็นไปได้ แม้ว่าไมโครซอฟท์บอกว่าพยายามพัฒนา COM เพื่อนำไปใช้บนระบบปฏิบัติการอื่นก็ตาม

ดังนั้นปัจจัยสำคัญที่ต้องพิจารณาเรื่องพอร์ตเทบิลิตี้ก็คือ แพลตฟอร์ม หากงานที่ทำมีโอกาสเกี่ยวข้องกับลูกค้าที่ใช้งานแพลตฟอร์มหลาย J2EE น่าจะเหมาะสมกว่าแต่ถ้าไม่ทั้ง J2EE และ .Net ก็ตัวเลือกที่ใกล้เคียงกัน ต้องพิจารณาปัจจัยอื่นด้วย



บทที่ 5

บทวิจารณ์และสรุป

5.1 บทวิจารณ์และสรุป

โครงการนี้ได้ทำการทดสอบการสร้างเว็บเซอร์วิส พร้อมทั้งการนำ WS- Security ไปใช้กับเว็บเซอร์วิส อีกทั้งทดสอบโดยการนำไปใช้ในระบบจริง ซึ่งผลการทดสอบ โดย Websphere Application Developer 5.1 กับ Microsoft Visual Studio .Net 2003 with WSE 1.0 ติดต่อกันได้ด้วยดี แต่อย่างไรก็ตาม ระหว่างการทดสอบเกิดปัญหามากมาย ซึ่งสรุปปัญหาต่างๆ ที่พบเจอได้เป็นข้อๆ ดังนี้

1. Key ที่นำมาใช้ต้องทำให้อยู่ในฟอร์แมตที่สามารถใช้ได้ เช่น ใน Java ให้ใช้คีย์ได้เฉพาะ JKS, JCEKS, PKCS12, CER และ ใน .Net ให้ใช้ได้เฉพาะ PFX ,PKCS12, CER ซึ่งหากนำคีย์มาใช้ผิดจะทำให้ไม่สามารถ implement WS-Security ได้

2. Tool ที่นำมาใช้พัฒนาควรตรวจสอบก่อนว่า Tool นั้น Support WS-Security หรือไม่ และ ใช้กับ Version อะไร ซึ่งปัญหาหลักๆ เกิดจากการใช้ Tool กับ Java ผิด Version ทำให้ไม่สามารถคอมไพล์ได้ หรือรันไม่ได้ เช่นในที่นี้หากนำ Websphere Application Developer 5.1 ซึ่ง Support J2EE 1.3 มาติดต่อโดยมีการ implement WS-Security กับ Rational Application Developer 6.0 หรือ Microsoft Visual Studio .Net 2003 with WSE2.0 ก็จะไม่สามารถติดต่อกันได้

3. ใน Tool ที่เกิดขึ้นมาใหม่จำนวนมาก หลาย tool ที่สนับสนุนการทำ Web service แต่ไม่ได้สนับสนุน WS-Security ซึ่งต้องตรวจสอบให้รอบคอบก่อนนำมาใช้ ซึ่งปัจจุบันมี มาตรฐานใหม่ๆ เกิดขึ้นมาเรื่อยๆ ซึ่งต้องพิจารณาให้ดีว่าสมควรไหมที่ต้องทำตามมาตรฐานใหม่นั้น ซึ่งบางอย่างไม่จำเป็น

4. การส่งข้อมูลระหว่าง Server กับ Client ในการติดต่อกับระหว่าง Web Service ข้าม Platform ระหว่าง Java กับ .Net นั้น ถ้าเป็นข้อมูลพื้นฐาน เช่น String Integer Boolean นั้นสามารถที่จะติดต่อกันได้ แต่ถ้าส่งข้อมูลที่มีความซับซ้อน หรือ ลักษณะเป็น Object ทั้งสองฝ่ายยังมีการประกาศการส่งข้อมูลแตกต่างกันอยู่ทำให้ไม่สามารถส่งข้อมูลประเภท Object ข้าม Platform

ปัญหาการส่งข้อมูลเป็น Set ระหว่าง .Net กับ Java

- การส่งข้อมูลจาก .Net ไปยัง Java : เซตของข้อมูลที่ .Net ส่งออกมาเรียกว่า Dataset ซึ่งเมื่อ return ค่าออกมาแล้วจะเป็น XML ซึ่งเมื่อ Java รับสามารถใช้ SAXParser รับและอ่านออกมาได้

- การส่งข้อมูลจาก Java ไปยัง .Net : เซตของข้อมูลของ Java ปกติจะใช้ Resultset ซึ่งเป็น Runtime Datatype ซึ่งไม่สามารถส่งออกไปข้ามเครื่องได้ จึงต้องแปลงเป็น XML String เพื่อใช้ในการ

ส่งข้อมูลเข้าไปยังเครื่องอื่น แล้วหากปลายทางต้องการรับต้องแปลงข้อมูล XML เพื่อส่งต่อ ในกรณีของ .Net สามารถประกาศ Dataset มารับได้เลย

ถึงแม้ว่าจะสามารถปฏิบัติได้ตรงตรงตามเป้าหมายที่วางไว้แต่เว็บเซอร์วิสในปัจจุบันก้าวหน้าไปอย่างรวดเร็ว จึงทำให้ขาดความรู้ใหม่ๆ ของเว็บเซอร์วิส อีกทั้งการมุ่งศึกษาด้าน WS-Security เพียงด้านเดียวก่อให้เกิดปัญหาในส่วนอื่นๆ ของเว็บเซอร์วิสอีกมากมายที่ยังไม่มีโอกาสได้ศึกษาเพิ่มเติม

5.2. แนวทางในการพัฒนาต่อ

จากการศึกษาเรื่องเว็บเซอร์วิส และ WS-Security แสดงให้เห็นว่าหากต้องการที่จะทำเว็บเซอร์วิสที่สามารถใช้ได้จริงต้อง มีการใช้ WS-Security เน้นอนการ implement ย่อมเป็นไปได้ยาก ในวิทยานิพนธ์ฉบับนี้ได้เขียนวิธีการ implement ไว้เพื่อให้ผู้สนใจสามารถนำไปพัฒนาต่อ ในด้านการนำไปประยุกต์กับเว็บเซอร์วิสใหม่ๆ ในอนาคต ได้ ส่วนหากมีผู้สนใจนำปริญญานิพนธ์ฉบับนี้ไปพัฒนาต่อสามารถนำไปใช้ในเรื่องของการศึกษามาตรฐานใหม่ๆ ของ WS-Security ซึ่งต้องทำความเข้าใจค่อนข้างมาก หรือการเขียน Tool ขึ้นมาสนับสนุนการสร้าง SOAP ให้ตรงกับ มาตรฐาน ของ WS-Security หรืออาจเขียนโปรแกรมสำหรับ Sign หรือ Encrypt SOAP ออกมาเป็นรูปแบบ GUI ของตัวเอง ซึ่งเป็นประโยชน์ต่อการสร้างเว็บเซอร์วิสอื่นๆ ต่อไป

บรรณานุกรม

- [1] IBM WebSphere Developer Technical Journal: Using Web Services Security in WebSphere Application Server
: http://www-128.ibm.com/developerworks/websphere/techjournal/0404_bose/0404_bose.html
- [2] Introduction to cryptography, Part 2: Symmetric cryptography
: <http://www-128.ibm.com/developerworks/security/library/s-crypt02.html>
- [3] Introduction to cryptography, Part 3: Asymmetric cryptography
: <http://www-128.ibm.com/developerworks/security/library/s-crypt03.html>
- [4] Exploring XML Encryption, Part 1
: <http://www-128.ibm.com/developerworks/xml/library/x-encrypt/>
- [5] Best Practices for Web services, Part 11: Web services security, Part 1
: <http://www-128.ibm.com/developerworks/webservices/library/ws-best11/>
- [6] Configure Web Services Security with WebSphere: Part 2, Digital Signatures and Encryption
: http://www128.ibm.com/developerworks/websphere/library/techarticles/0505_cowan/0505_cowan.html
- [7] Building secure Web services with WebSphere Studio: Part 1: XML signature
: http://www-128.ibm.com/developerworks/websphere/library/techarticles/0403_liu/0403_liu.html
- [8] JavaTM Cryptography Architecture API Specification & Reference
: <http://java.sun.com/j2se/1.5.0/docs/guide/security/CryptoSpec.html#JceKeystore#JceKeystore>
- [9] Developer Forums-Java Forums - importing pfx certificate WITH private key into JKS keystore?
: <http://forum.java.sun.com/thread.jspa?forumID=2&threadID=344200>
- [10] RSA Security
: <http://www.rsasecurity.com/node.asp?id=1366>
- [11] keytool - Key and Certificate Management Tool
: <http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/keytool.html#CertChains>
- [12] Convert the Java JKS key-store to Microsoft PFX format
: <http://www.cronics.com/products/opensource/faq/signFree.htm>
- [13] Generating a Keystore
: http://www.omii.ac.uk/docs/2.3.1/security/non-omii/ukes_keystore_generation.htm

[14] KeyStore Explorer

: <http://www.lazgosoftware.com/kse/>

[15] JaBaCATs: Java Basic Certificate Authority Tools

: <http://www.jabacats.com/>

[16] Establishing WSRP Security

: <http://edocs.beasys.com/wlp/docs81/wsrp/security.html>

[17] Java Toolkit

: <http://www.syncedit.com/software/javatoolkit/>

[18] Internet Explorer Was Unable to Import This Certificate

: <http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/Q182/0/54.ASP&NoWebContent=1>

[19] comp.lang.java.security (.NET WSE and Java security WS-Security interop)

: http://groups.google.co.th/group/comp.lang.java.security/browse_thread/thread/ae5125cff5d332f59450aeaf0ea26f#59450aeaf0ea26f

[20] OpenSSL

: <http://www.openssl.org/about/>

[21] Web Services and other distribute technologies:WS-Security Interoperability Using WSE 2.0 and JWSDP 1.4

: <http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnbda/html/interopsun.asp>

[22] Build secure Web services

: <https://www6.software.ibm.com/developerworks/education/r-secureradws/>

[23] Developing a Web service from a Java bean

: <http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp>

[24] Secure Web services: Interoperability

: <http://www-128.ibm.com/developerworks/edu/ws-dw-ws-secint-i.html>

[25] Security information Java technology Security

: [http://www-](http://www-128.ibm.com/developerworks/java/jdk/security/142/secguides/jssedocs/JSSERefGuide.html)

[128.ibm.com/developerworks/java/jdk/security/142/secguides/jssedocs/JSSERefGuide.html](http://www-128.ibm.com/developerworks/java/jdk/security/142/secguides/jssedocs/JSSERefGuide.html)

[26] org.mortbay.util Class PKCS12Import

:<http://jetty.mortbay.org/javadoc/org/mortbay/util/PKCS12Import.html>

[27] Java and .NET - AES Crypto Interop

: <http://blogs.msdn.com/dotnetinterop/archive/2005/01/24/359898.aspx>

[28] MS Keynote

: <http://trevor.smith.name/MoreETConTexts/MS%20Keynote.txt>

[29] Web Services Enhancements 1.0 and Java Interoperability, Part 2

: <http://msdn.microsoft.com/architecture/solvenow/solveintsys/default.aspx?pull=/library/en-us/dnwse/html/wsejavainterop2.asp>

[30] IBM developerWorks : Web Services : Education - Tutorials Digital signature: .Net client

: <https://www6.software.ibm.com/developerworks/education/ws-secint/ws-secint-7-1.html>

[31] Contract First Web Services Interoperability between Microsoft .NET and IBM WebSphere

: <http://msdn.microsoft.com/vstudio/java/interop/websphereinterop/default.aspx>

[32] Web Services Interoperability Guidance (WSIG): IBM WebSphere Application Developer 5.1.2

: <http://msdn.microsoft.com/webservices/webservices/building/interop/default.aspx?pull=/library/en-us/dnbda/html/wsinteroprecsibm-final.asp>

[33] WebSphere and .NET Web services security interoperability

: <http://www-128.ibm.com/developerworks/webservices/library/ws-was-net/>

[34] WS-Security Interoperability Using WSE 2.0 and Sun JWSDP 1.5

: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/wssinteropjwsdp15.asp>

[35] Secure Web services:Encryption (Introduction)

: C:\Documents and Settings\Administrator\Desktop\ws-encrypt\ws-encrypt\ws-encrypt-1-2.html